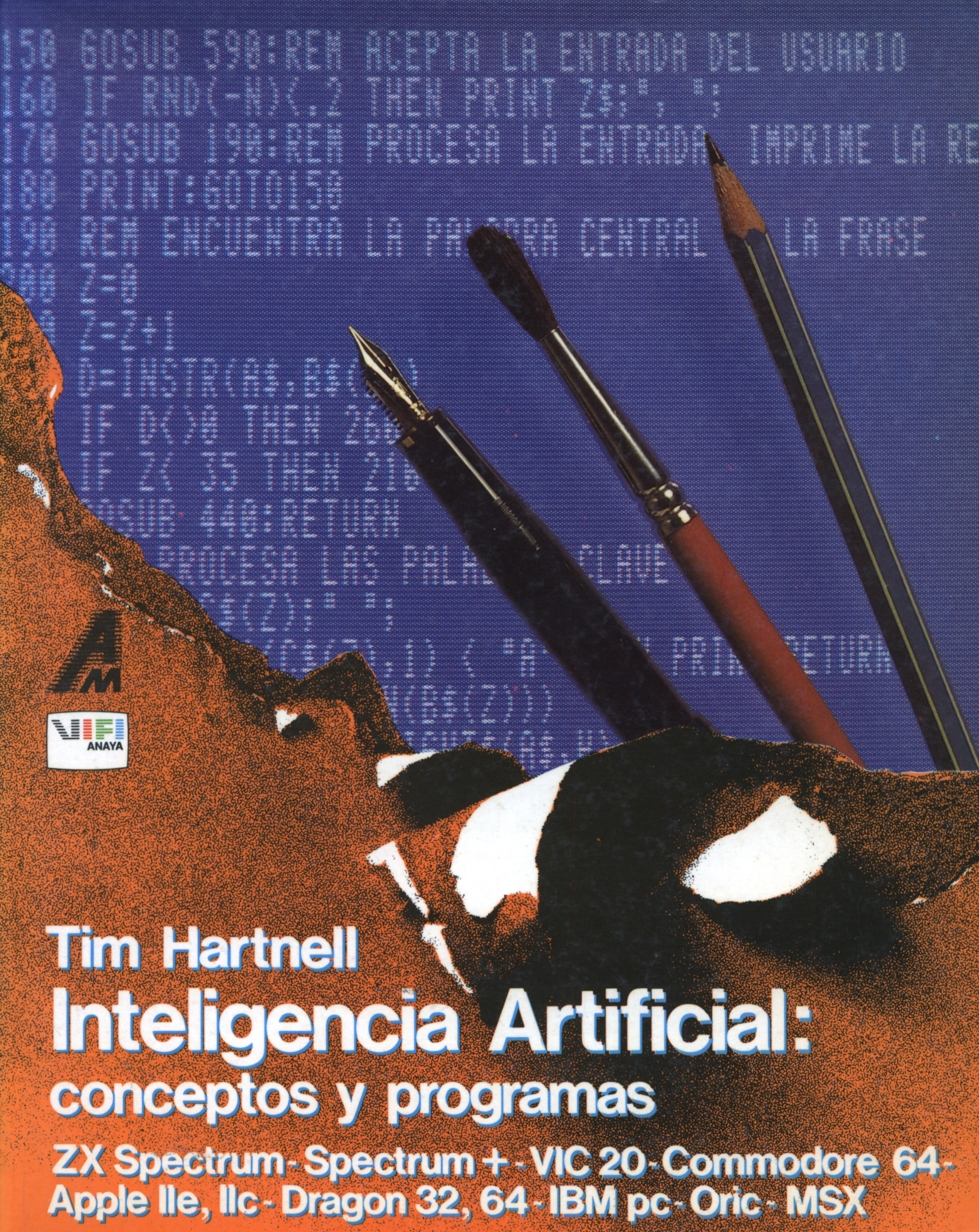

150 GOSUB 590:REM ACEPTA LA ENTRADA DEL USUARIO
160 IF RND(-N)<.2 THEN PRINT Z$;", ";
180 PRINT:GOTO150
VIFI
ANAYA
Tim Hartnell
Inteligencia Artificial:
conceptos y programas
ZX Spectrum - Spectrum + - VIC 20 - Commodore 64 -
Apple IIe, IIc - Dragon 32, 64 - IBM pc - Oric - MSX

INTELIGENCIA ARTIFICIAL

Conceptos y programas

Inteligencia artificial

Conceptos y programas

Tim Hartnell



ANAYA MULTIMEDIA

MICROINFORMATICA

Título de la obra original:
EXPLORING ARTIFICIAL INTELLIGENCE ON YOUR MICROCOMPUTER

Traducción de: Pilar Vázquez

Diseño de colección: Antonio Lax

Diseño de la cubierta: Narcís Fernández

Publicado por primera vez en
Gran Bretaña por INTERFACE PUBLICATIONS

Villafranca, 22. 28028 Madrid
Depósito legal: M. 8.165-1985
ISBN: 84-7614-014-2
Printed in Spain
Imprime: JOSMAR, S. A.
Artesanía, 17. Políg. Ind. de COSLADA (MADRID)

Indice

Sección segunda: LA INVESTIGACION

Sección tercera: EL HABLA

Sección cuarta: LA AYUDA

Apéndices

Todos los listados son compatibles, tanto con el lenguaje BASIC Microsoft como con el BASIC MSX, y se pueden ejecutar en la mayoría de los microordenadores, incluyendo el Commodore 64, el BBC, el VIC 20 (algunos programas exigirán memoria extra), el Dragon 32, el Oric, el TI 99/4A con BASIC ampliado, el TRS-80 (en todos sus modelos, aunque se necesitará memoria adicional en el MC-10, en el caso de algunos programas), el Apple II, el IBM PC y muchos otros más. La manipulación de cadenas es la estándar del Microsoft, de modo que algunos ordenadores necesitarán ciertas modificaciones en este sentido. El Apéndice 4 contiene los listados para el ZX Spectrum y Spectrum +.

Prefacio

Estás a punto de lanzarte, querido lector, a una aventura fascinante en un reino en donde se difuminan las barreras entre el hecho científico y la ciencia ficción.

Desde que salieron a la luz los primeros ordenadores, se ha venido librando un continuo debate, angustiado a veces, en torno a temas tales como:

— ¿Puede la máquina pensar realmente?
— ¿Cuál es la naturaleza de la inteligencia?, y en relación con esto, ¿se llegará alguna vez a construir una máquina que pueda participar de esa inteligencia?

Cuando hayas hecho tuyo este libro, te sentirás capacitado para tomar parte activa en el debate y con autoridad para hacerlo, además. Porque en este libro vamos a investigar el mundo apasionante de la inteligencia artificial repitiendo algunos de sus programas más famosos.

Cubriremos gran parte del campo, desde los programas que aprenden y razonan a aquellos que te hablan, te obedecen o te aconsejan.

La escritura de este libro, ha sido una experiencia fascinante. El leer la extensa bibliografía existente, el llegar a familiarizarme con las aspiraciones de los pioneros de la inteligencia artificial y el escribir programas que —admitámoslo con toda crudeza— permitían repetir en un miniordenador algunos de sus descubrimientos más importantes, resultó ser un ejercicio extremadamente interesante y placentero.

Espero que este libro transmita al menos algo de la fascinación que yo experimenté y que tú, lector, sientas la misma emoción que yo sentí al ejecutar los programas.

Tim Hartnell
Londres, 1984

Algunas indicaciones previas: Todos los programas han sido escritos utilizando un subconjunto del lenguaje BASIC que es compatible tanto con el BASIC Microsoft, como con el BASIC MSX. Los programas utilizan órdenes estándar Microsoft para la manipulación de cadenas alfanuméricas (MID$, RIGHT$, LEFT$) que no tienen soporte en las máquinas SINCLAIR, sin embargo encontrarás los listados completos para el ZX Spectrum en el Apéndice 4. Unos cuantos programas utilizan una variable llamada FACTORES. Si tu sistema no acepta las palabras reservadas (como OR en este caso) que aparecen en nombres variables, cámbialo a FACT o algo por el estilo. Sólo las dos primeras letras de un nombre variable son importantes, de modo que si quieres puedes omitir el resto.



Sección primera: EL PENSAMIENTO

1
Aprendizaje y razonamiento

El continuo debate en el que nos encontramos inmersos en la actualidad gira en torno a la cuestión de saber si por el hecho de estar produciendo máquinas que pueden comportarse de un modo que parece inteligente, nos estamos acercando realmente a la producción de inteligencia. Inextricablemente ligada a este debate hay otra cuestión que atañe a la naturaleza de la inteligencia.

Es cierto que los programas incluidos en este libro permiten al ordenador dar muestras de inteligencia respondiendo a situaciones, tomando decisiones y actuando de acuerdo con ellas. Pero, sin embargo, no se sugiere en ningún momento que el ordenador sea consciente de sus actos. No se echa a reír ante las erróneas conclusiones a que se llega en DOCTOR y no puede admirar —ni siquiera reconocer— los poemas producidos por HANSHAN ni aun cuando le salen particularmente efectistas.

¿Estamos, pues, justificados cuando afirmamos que estamos produciendo "inteligencia artificial"? Yo creo que sin el tipo de percepción que reconoce cosas tales como el "efectismo" de un poema o la incongruencia de una respuesta, no podemos sugerir que nos encontremos ante ningún tipo de inteligencia.

La inteligencia artificial está dando sus primeros pasos y pretender provocar una conciencia y una percepción reales a partir de un breve programa BASIC ejecutado en un miniordenador, cuando las gigantescas máquinas de estructura principal no han llegado todavía a arañar la superficie de este campo, es bastante poco realista.

Hay, sin embargo, dos áreas de comportamiento, razonables candidatas ambas para clasificar un comportamiento como inteligente, que pueden obtenerse en cualquier ordenador. Se trata de los campos del aprendizaje y el razonamiento.

TICTAC, un programa que juega a algo parecido al TRES EN RAYA, empieza su vida sabiendo tan sólo cómo se gana y cómo se obstaculiza al contrario para que no pue-

da mover. Carece de todo conocimiento relativo a qué jugadas ha de hacer primero con el fin de que sus posibilidades de ganar sean mayores. En realidad, su conocimiento inicial es tan escaso que su juego no podría ser peor.

Pero, enfrentado a un oponente que juegue totalmente al azar (un oponente que ni siquiera sepa que gana la partida el que primero consiga poner tres de sus fichas en raya), en diez partidas más o menos TICTAC habrá aprendido el valor de mover, siempre que le sea posible, a la casilla central y habrá ordenado el resto de sus jugadas en una secuencia que —si bien difiere de la secuencia que tú, lector, o yo crearíamos en circunstancias similares— le permite ganar una proporción cada vez más elevada de juegos, incluso contra un contrincante inteligente, como puede serlo el lector. TICTAC fue escrito con la intención de mostrar el estado de su aprendizaje después de cada partida. Esto hace que la ejecución de este programa sea fascinante. Por otro lado, ofrece la posibilidad de desarrollarlo de muchas maneras diferentes con el fin de investigar su capacidad de aprendizaje.

SILOGISMOS es nuestro programa de razonamiento. Su objetivo es resolver silogismos del tipo de aquellos que todos conocemos, como:

SOCRATES ES UN HOMBRE
TODOS LOS HOMBRES SON MORTALES
LUEGO, SOCRATES ES MORTAL

Partiendo de las dos premisas iniciales, SILOGISMOS extrae una conclusión. Lo que es importante tener en cuenta en este programa es que SILOGISMOS puede llegar a conclusiones sobre una información que no le ha sido suministrada explícitamente.

Explicaré esto. Observemos estas dos premisas:

UNA NOVELA ES UN LIBRO
UN LIBRO SE IMPRIME EN PAPEL

Aunque no se le ha dicho de modo explícito al programa que una novela se imprime en papel, responderá SI cuando se le pregunte:

¿SE IMPRIME EN PAPEL UNA NOVELA?

El lector puede pasárselo muy bien introduciendo en el programa una amplia variedad de premisas y haciéndole después todo tipo de preguntas sobre ellas para ver qué conclusiones puede sacar SILOGISMOS. NO TENGO DATOS, NO y NO SE se encuentran también entre las posibles respuestas de SILOGISMOS.

Ya en los primeros momentos del debate sobre la inteligencia de las máquinas ("¿pueden las máquinas llegar a ser verdaderamente inteligentes?") quedó claro que había que examinar detenida y minuciosamente los términos fundamentales que se estaban sometiendo a debate. ¿Qué queríamos decir realmente por pensamiento? Si no sabíamos de verdad lo que queríamos decir cuando usábamos este término referido a nosotros mismos, ¿cómo podíamos juzgar el rendimiento de las máquinas en este campo?

Este tipo de reflexión es uno de los muchos efectos que ha tenido el estudio de la inteligencia artificial. El hombre se ha visto forzado a reflexionar sobre sí mismo y a examinar áreas del comportamiento humano de un modo que muy pocos hombres hasta ahora se habían preocupado de hacerlo.

Unas líneas más arriba sugería que aunque es cierto que las máquinas no llegan siquiera a rozar el tipo de conciencia que es un prerrequisito vital para afirmar que un sistema es inteligente, algunos aspectos de la inteligencia —el razonamiento y la capacidad para aprender— se encontraban, no obstante, entre nuestras capacidades presentes.

Hay diferentes tipos de aprendizaje. Podemos aprender observando a los otros, leyendo, por lo que los otros nos dicen (que es un tipo de "lectura verbal", de modo que los dos están muy relacionados) y "cometiendo errores". Los ordenadores pueden aprender de todas estas maneras. TICTAC aprende en gran medida cometiendo errores, aunque tiene cierto conocimiento preprogramado (que aumenta por medio de lo que le decimos).

Retroalimentación (Feedback)

Los errores de TICTAC carecerían, por supuesto, de sentido, si no volviera a recibir información *(feedback)* sobre sus logros o, en cualquier caso, sobre sus esfuerzos. La retroalimentación es un elemento vital en el aprendizaje.

Una de las primeras máquinas "capaces de aprender" fue la tortuga, precursora de una multitud de galápagos robóticos similares, construida en 1948 por Grey Walter, un fisiólogo especializado en el cerebro. Construyó su tortuga —media esfera que se arrastraba por el suelo abriéndose paso entre los obstáculos y que se iba a la cama cuando se empezaba a quedar sin pilas— para demostrar su tesis de que el comportamiento complejo, independientemente de lo complicado que pueda parecerle al observador ajeno, está basado en la interacción de tan sólo unas cuantas ideas básicas.

La tortuga aprendía el camino utilizando *feedback* negativo, es decir, tendía a no repetir un comportamiento que no fuera productivo. Una tortuga que no aprendiera que el chocar repetidamente contra una pared no es la mejor manera de avanzar, no llegaría muy lejos.

¿Cómo piensan las máquinas?

Los ordenadores actuales son procesadores en serie. Es decir, avanzan de un punto a otro, paso a paso, estando los pasos futuros determinados por los resultados de los presentes. El cerebro humano, por el contrario, utiliza no sólo el procesado en serie, sino también el procesado paralelo en el cual varias corrientes de pensamiento —conscientes unas e inconscientes otras— progresan a la vez.

El proceso de pensamiento y toma de decisiones de un ordenador es en esencia un sendero a través de un intrincado laberinto de construcciones IF/THEN (SI/ENTONCES):

IF esto es cierto AND esto es cierto
AND esto no es cierto THEN haz esto

El ordenador, por supuesto, puede tomar decisiones 'OR' del mismo modo que decisiones 'AND':

IF esto es cierto OR esto es cierto
THEN haz esto

Pueden combinarse:

IF esto es cierto AND esto es cierto OR
algo más es cierto THEN haz esto

¿Cómo lo hace el ordenador? El primer mecanismo electrónico para calcular fue construido (en la cocina de su casa) por George Stibitz, que trabajaba en los Bell Telephone Laboratories, en la década de los años cuarenta. Este conectó pilas, bombillas y algunos interruptores de teléfono con relés para calcular en términos binarios. (Este es el sistema de numeración que tiene sólo dos dígitos: el 0 y el 1. Cuando un interruptor estaba encendido se consideraba que estaba en 1, mientras que cuando estaba apagado se le consideraba igual a 0.) Stibitz se dio cuenta de que este tosco mecanismo, suficientemente desarrollado, podría servir para todo tipo de problemas matemáticos. (De lo que, al parecer, no se dio cuenta —como el lector verá enseguida— es de que los mismos circuitos que estaba utilizando para sumar números binarios podían utilizarse para lograr decisiones.)

No obstante, unos años antes, en 1937, Claude Shannon (que más tarde trabajaría también en los Bell) se había doctorado con una tesis sobre la relación entre el Algebra Booleana y el paso de la corriente eléctrica a través de circuitos orientados.

El Algebra Booleana —que es donde la parte "pensante" de la máquina comienza verdaderamente— se basa en los trabajos realizados por George Boole, profesor del Queens College de Cork, a mediados del siglo XIX. Su libro *Investigación de las leyes del pensamiento sobre las que se basan las teorías matemáticas de la lógica y la probabilidad* (publicado en 1854) puso los cimientos de la lógica simbólica moderna. El Algebra Booleana está basada en las leyes establecidas por Boole y es el pivote en torno al que gira la capacidad de razonar del ordenador.

Boole escribía en el prefacio de su libro:

> «Las leyes que tenemos que examinar son las leyes de una de nuestras facultades mentales más importantes. Las matemáticas que hemos de construir son las matemáticas del intelecto humano.»

Hasta el descubrimiento de Boole siempre se había dado por supuesto que la lógica era una rama de la filosofía. Boole demostró con toda claridad que ésta, sin lugar a dudas, pertenecía al reino de las matemáticas.

Interruptores y decisiones

Investigaremos ahora las afirmaciones de Boole y veremos cómo se relacionan con el ordenador, el mecanismo de toma de decisiones y la inteligencia artificial, mediante la

reconstrucción mental de algunos de los mecanismos que construyó Stibitz en la cocina de su casa. Empezaremos con un circuito muy simple que contiene un generador de corriente eléctrica (pila), un interruptor simple y una bombilla.

INTERRUPTOR —— BOMBILLA

PILA

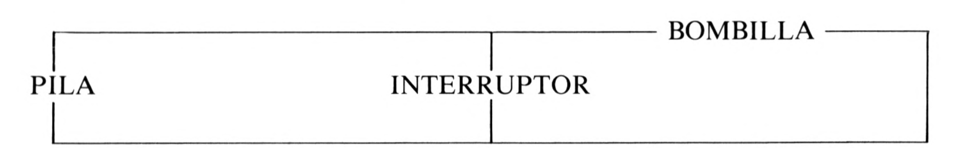
Se puede ver que cuando el interruptor está cerrado, pasará la corriente y la bombilla se encenderá. Indicaremos que el interruptor ha sido conectado diciendo que su estado es '1'. Cuando el interruptor no está conectado, y la corriente no pasa, se dice que su estado es '0'. Igualdad 1, desigualdad 0. Posteriormente, adoptaremos la convención de que cuando la bombilla está encendida, su estado es 1; cuando está apagada, su estado es 0.

Este circuito se denomina circuito de AFIRMACION. Cuando el interruptor está encendido, la luz está encendida. Es decir, el estado del interruptor es igual al estado de la luz. Si dibujáramos un pequeño gráfico que mostrara la relación entre el estado de la bombilla y el del interruptor en un circuito de AFIRMACION *(ASSERTION)* nos saldría algo así:

INTERRUPTOR	BOMBILLA
0	0
1	1

Señalaremos, de paso, que este gráfico recibe el nombre de "tabla de verdad".

Veremos ahora otro circuito simple:

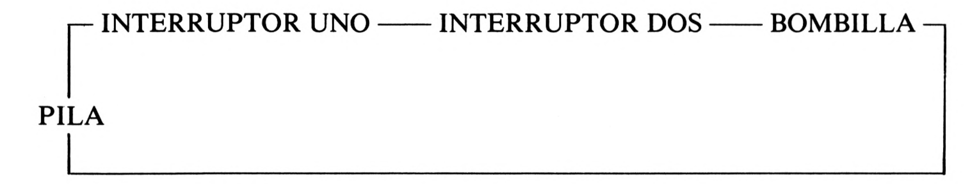


Si se observa, se verá que la bombilla está encendida (estado de la bombilla igual a 1) cuando el interruptor está abierto (el estado del interruptor es 0) y —en el momento en que se cierra el interruptor (interruptor puesto a 0)— la corriente pasará a través suyo, en vez de a través de la bombilla.

Este es un circuito de NEGACION, y la tabla de verdad de este circuito sería la siguiente:

INTERRUPTOR	BOMBILLA
0	1
1	0

Llegamos ahora a la parte interesante, en donde los circuitos pueden "tomar decisiones". Imaginemos que tenemos un circuito con dos interruptores, como el siguiente:

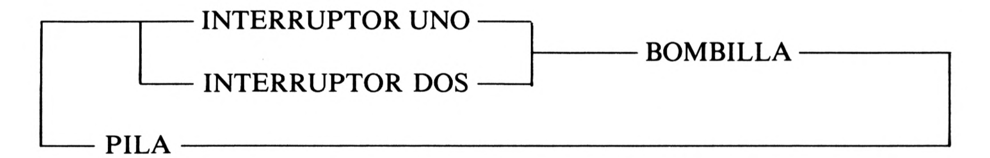


Con los dos interruptores cerrados (es decir, encendidos, siendo el estado de ambos igual a 1 [1 1]) la bombilla lucirá. Si uno de los dos interruptores está apagado (un interruptor puesto en 1, y el otro en 0 [1 0]) o los dos están apagados (el interruptor uno puesto a 0 y el interruptor dos igual a 0 [0 0]) la bombilla estará apagada. Este circuito se denomina "circuito de puerta 'AND' ".

La tabla de verdad tendría este aspecto:

INTERRUPTOR UNO	INTERRUPTOR DOS	BOMBILLA
0	0	0
1	0	0
0	1	0
1	1	1

De 'AND' pasamos a 'OR'. El "circuito de puerta 'OR'" es así:



En este circuito que tiene los interruptores paralelos (en el circuito 'AND' eran lineales), la bombilla se encenderá (estado 1) si uno de los dos interruptores, uno o dos, está en uno (ya sea [0 1] o [1 0]) o los dos interruptores están en uno [1 1]. Antes de continuar leyendo, intenta, lector, construir por ti mismo la tabla de verdad del "circuito de puerta 'OR' ".

INTERRUPTOR UNO	INTERRUPTOR DOS	BOMBILLA
0	0	0
1	0	1
0	1	1
1	1	1

Puertas lógicas "reales"

El ordenador utiliza puertas lógicas exactamente iguales que estos circuitos, aunque, por supuesto, no se trata de esos grandísimos interruptores que hay que encender y apagar. Una de las razones por las que Shannon y Stibitz usaron interruptores con relé es que éstos pueden encenderse sin tener que tocarlos (al aplicar una corriente eléctrica se genera una fuerza magnética que cierra el interruptor).

En el ordenador tampoco hay relés eléctricos del tipo de Shannon, aunque los elementos de los "chips" de éste actúan como miles y miles de relés. En un diagrama esquemático de los circuitos, las puertas que hemos examinado unas líneas más arriba aparecen como sigue:

En primer lugar hay un "inversor". Si una señal al entrar tiene el estado 0, cuando sale del mecanismo tiene el estado 1, y a la inversa:

A ENTRADA (INPUT) — B SALIDA (OUTPUT)

A es igual al contrario de B

Esta es una puerta lógica 'AND':

A, B — C

C = A*B

Y esta es una puerta 'OR':

A, B — C

C = A+B

Hay otra puerta que se suele utilizar bastante a menudo en los circuitos y que es útil cuando uno está intentando calcular la manera en que los circuitos llegan a tomar decisiones. Se trata de la puerta 'XOR', la 'OR EXCLUSIVA'. Con ésta, cuando cualquier entrada es 1 ([1 0] o [O 1]) el estado de la salida es 1. No obstante, si las dos son 1 [1 1], o las dos son 0 [0 0], el estado de la salida es 0.

Aquí tenemos un diagrama esquemático de la puerta 'XOR':

A, B — C

C = A⊕B

Y esta es la tabla de verdad de la puerta 'XOR':

INTERRUPTOR UNO (A)	INTERRUPTOR DOS (B)	BOMBILLA (C)
0	0	0
0	1	1
1	0	1
1	1	0

Tan sólo con estos elementos se pueden construir "circuitos" y conseguir decisiones. Es fácil escoger, por ejemplo, de entre las entradas que hemos examinado (y yo sugiero al lector que intente encontrar él mismo cuáles son) una secuencia de entradas que represente sentencias como las que siguen:

IF A AND B son ciertas (es decir, [1 1]) AND C OR
D (pero no las dos) es cierta ([1 0] o [0 1]
pero no [1 1] y ciertamente no [0 0])
THEN D es cierta (es decir, la salida es 1).

La tarea de calcular cómo han de combinarse los interruptores para imitar una sentencia como ésta —o similares— es fascinante, al tiempo que uno puede hacerse una idea de la manera en que una simple secuencia de operaciones booleanas elabora las decisiones para llegar a los resultados (si este circuito estuviera de hecho correctamente construido, se encendería una bombilla en D). El realizar este tipo de operaciones ayuda a comprender un poco mejor el funcionamiento del ordenador. Partiendo de esta comprensión uno se hace realmente consciente de lo complicadas que pueden llegar a ser estas cosas, de la complejidad que requieren cuando se pretende estimular un tipo de comportamiento "inteligente" por encima del más básico.

Consideremos, por ejemplo, el sistema de circuitos necesarios para emular el funcionamiento del ordenador en el primer programa de esta sección, TICTAC.



2
Un programa que aprende

Muchos programas de inteligencia artificial no entran en el ordenador totalmente formados. Aun cuando ya se le han corregido los errores y están siendo ejecutados, les falta todavía mucho para estar totalmente acabados. El programa que observaremos en este capítulo, TICTAC, una versión del TRES EN RAYA, es uno de esos programas "sin formar". TICTAC aprende a medida que juega, modificando sus reglas a la luz de los éxitos que vaya obteniendo o, de no ser así, de su comportamiento más reciente.

En un programa que va a aprender a medida que es ejecutado, las normas de funcionamiento han de adoptar una forma "blanda" que pueda modificarse a medida que el juego evoluciona. En este programa el ordenador conoce las reglas del juego y cuenta con una sección específica para, por un lado, bloquear a su contrincante cada vez que éste intente hacer tres en raya y, por el otro, para formar él mismo tres en raya cuando se le presente la oportunidad, pero al principio carece totalmente de una estrategia de juego.

Este es el esquema del tablero de TICTAC:

```
1 : 2 : 3
-----------
4 : 5 : 6
-----------
7 : 8 : 9
```

El programa juega seleccionando casillas de acuerdo con una secuencia que él mismo desarrolla a medida que se van sucediendo las partidas. Si gana la partida, mueve las posiciones escogidas aproximándolas a la cabeza de la secuencia. No hace ningún cambio en el caso de que la partida haya quedado empatada. Una derrota provoca el desbarata-

miento de la secuencia, con lo que es muy poco probable que sean esas mismas las jugadas que escoja en la siguiente partida.

Tú y yo sabemos, lector, que la casilla central (la cinco en el diagrama que hemos visto) es la que hay que ocupar siempre que esté vacía. Al principio TICTAC no lo sabe. De hecho, se le ha dado deliberadamente una mala baza de salida —siendo la casilla dos la primera de sus posibilidades— con el fin de que nos sea más fácil ver el efecto que tiene el aprendizaje sobre su juego.

Al final, si el mecanismo de aprendizaje ha funcionado debidamente, TICTAC debería darse cuenta de que, siempre que esté vacía, es una buena idea hacerse con la casilla o posición cinco. En realidad, como veremos más adelante, TICTAC sí que llega, de hecho, a esta conclusión incluso jugando contra un contrincante que no sepa en absoluto de qué va el juego y que, por tanto, juegue totalmente al azar. Es razonable, pues, suponer que si TICTAC jugara contra un contrincante inteligente —como el lector— su juego mejoraría aún más rápidamente.

Donald Michie, un pionero de la investigación en el campo de la inteligencia artificial de la Universidad de Edinburgo, todavía hoy una eminencia, investigó el "aprendizaje automático" en el juego del TRES EN RAYA. Utilizó un mecanismo denominado "cajas" en el que una meta se divide en varias submetas. Cada caja que se forme contiene la información de una submeta.

La meta del TRES EN RAYA es ganar. Cada submeta consiste en hacer, en primer lugar: *a*) una jugada correcta y, finalmente, *b*) la mejor jugada dada la situación de la partida.

Michie calculó que fundamentalmente son 288 las jugadas o posiciones diferentes que tiene ante sí un jugador del TRES EN RAYA. Procedió entonces a construir, como a continuación se indica, su contrincante mecánico (un experimento que quizá al lector le gustaría repetir). Michie tomó 288 cajas de cerillas y pintó encima de cada una de ellas un tablero con una de estas jugadas o posiciones diferentes y con las casillas vacías numeradas por orden. Acto seguido escribió en unos papelitos los números escritos en las casillas vacías. Cada número estaba repetido varias veces con la misma cantidad de cada número por caja. Es decir, si las casillas tres y cuatro estaban vacías en una posición determinada, la caja de cerillas había de contener, pongamos, cinco pedacitos de papel con el número tres escrito en ellos y cinco con el número cuatro.

Jugaba del modo siguiente. La primera jugada se hacía abriendo la caja que tuviera el tablero en blanco (sin jugada, sólo las casillas). Dentro de esta caja tenía que haber, entonces, cinco trocitos de papel por cada uno de los números del uno al nueve. Escogía al azar uno de los papelitos y movía a la casilla que se le indicaba en éste. Michie iba anotando en una lista los números que salían y de qué caja.

Al final del juego, Michie volvía a esta lista. Si el "ordenador-caja de cerillas" había ganado la partida, se añadía en cada una de las cajas apuntadas en la lista (las utilizadas en esa partida) otro papelito con el mismo número que hubiera salido en cada caso. Es decir, si la primera caja usada, la que tenía el tablero en blanco, proporcionaba el número cinco, se metía en ésta otro trocito de papel con el número cinco. Naturalmente, esto aumentaba las posibilidades de que volviera a salir el cinco la siguiente vez que se abriera esa caja para escoger un número al azar. El proceso continuaba hasta haber hecho lo propio en cada una de las cajas usadas.

Si la partida acababa en empate, no se modificaba el contenido de las cajas. Si el "ordenador" perdía la partida, se sacaban de las cajas utilizadas los papelitos que

habían originado las jugadas de esa partida perdida, reduciendo así las posibilidades de que volvieran a salir esos mismos números cuando se abrieran sus cajas respectivas (es decir, la siguiente vez que el ordenador se encontrara ante las mismas situaciones de tablero dibujadas en esas cajas).

En un trabajo publicado en 1968, "Boxes: An Experiment in Adaptive Control" (Chambers, R. A., and Michie, D., *Machine Intelligence 2* [Ed. Dale, E., and Michie, D.], Oliver and Boyd, 1968, pp. 137-152), Michie explica que las cajas "aprendieron" tan bien que, tras haber jugado 1.000 partidas contra un contrincante que jugaba totalmente al azar, el programa ganaba entre un 75 y un 87 por 100 de todas las partidas jugadas. En el caso de TICTAC no se puede esperar una proporción similar (aun cuando uno tenga la paciencia de jugar 1.000 partidas contra él), pero con todo, lo hace extremadamente bien, sobre todo si se cuentan los empates y se le ofrecen al programa las oportunidades necesarias para aprender.

Samuel y el tablero de damas

Las "cajas de cerillas inteligentes" de Michie no eran más que un juguete comparadas al programa de damas creado, a finales de los sesenta, por Arthur Samuel, de IBM. Lo que vamos a comentar aquí es uno de sus últimos programas, tal como aparece esbozado en el trabajo "Some Studies in Machine Learning Using the Game of Checkers —II— Recent Progress" (Samuel, A., *IBM Journal of Research and Development,* volumen 11 [noviembre 1967], pp. 601-617). Es interesante observar, sin embargo, que el programa final, por todos aclamado, no salió de su cabeza con toda su majestad, totalmente formado.

En realidad, Samuel había empezado programando juegos de damas en 1952 trabajando en el para entonces poderoso IBM 701. Dos años más tarde pasó el programa a un IBM 704 y en 1955 empezó a desarrollar la capacidad de aprendizaje de éste. El programa tenía en cuenta unos 40 factores en el momento de pensar la jugada, aunque para calcular una jugada en concreto se utilizaban menos de la mitad. Sabía cuándo un determinado factor no aportaba ninguna información para la elección de la jugada que había de hacer en ese momento y entonces lo olvidaba temporalmente.

El número de fichas con que contaba cada jugador era una cuestión importante y el programa de Samuel (como la mayoría de los que le siguieron) intercambiaba con toda alegría sus fichas cuando tenía más que su contrincante, pero se volvía muy avaro a este respecto cuando iba perdiendo, desde un punto de vista material. Entre los otros factores que tenía en cuenta el programa cuando evaluaba sus fuerzas, se incluyen el control del centro del tablero y el número de fichas del contrincante que se podían atacar con una sola jugada.

Un poco más adelante examinaremos más detenidamente los aspectos de la inteligencia artificial en los juegos de tablero (con el juego RISITAS inventado precisamente para este libro), pero de momento, el interés primordial del programa de Samuel reside en su capacidad para aprender. CHECKERS tenía dos maneras de aprender: mediante la rutina o mediante la automodificación.

Mediante la rutina, el programa almacenaba los resultados de sus investigaciones en relación con las jugadas posibles en la última situación del tablero. Esto significaba que la siguiente vez que se encontrara con esa situación, el programa no tendría que recorrer

de nuevo todo el proceso de calcular sus implicaciones; el resultado ya estaba allí. Este método, por supuesto, come mucha memoria, pero es muy efectivo. Al final el programa llegó a jugar casi a nivel de campeonato y había "recordado" prácticamente todas las posiciones de tablero que valían la pena.

Decíamos un poco más arriba que la función de evaluación de Samuel hacía uso de 40 factores. El proceso de automodificación funcionaba del modo siguiente. Samuel permitía al programa que investigara adelantándose a la posición que tuviera en ese momento y que llegara a una conclusión en cuanto al valor de ciertas jugadas y posiciónes. El programa utilizaba, asimismo, su función de evaluación para llegar a una conclusión partiendo de la misma posición en el tablero.

Samuel argumentaba que, si la función de evaluación era perfecta, generaría el mismo consejo que el mecanismo de investigar hacia adelante. Los factores de la función de evaluación se modificaban después de cada jugada a la luz de la diferencia entre el hallazgo de la investigación hacia adelante y la información dada por la función de evaluación. El trabajar de este modo suprimía el tener que estar dependiendo de una gran cantidad de memoria, como sucedía en el caso del aprendizaje mediante la rutina. Nuestro programa TICTAC no aprende del mismo modo que CHECKERS, pero su método implica automodificación más que acopio de información mediante la rutina.

El programa de TICTAC

El programa empieza con la siguiente subrutina de inicialización:

```
1180 REM INICIALIZACION
1190 CLS
1200 DIM A(9): REM TABLERO
1210 DIM M(10): REM ACCESO A LA BASE DE DATOS
1220 DIM W(24): REM DATOS DE VICTORIA O BLOQUEO
1230 DIM D(5): REM ACCESO AL MOVIMIENTO EN EL
     JUEGO ACTUAL
1240 REM DATOS DE VICTORIA O BLOQUEO
1250 FOR J = 1 TO 24
1260 READ W(J)
1270 NEXT J
1280 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
1290 DATA 1, 4, 7, 2, 5, 8, 3, 6, 9
1300 DATA 1, 5, 9, 3, 5, 7
1310 REM BASE INICIAL DE DATOS
1320 FOR J = 1 TO 10
1330 READ M(J)
1340 NEXT J
1350 DATA 2, 6, 8, 4, 7, 3, 1, 9, 5, 2
1360 RETURN
```

Se dimensionan cuatro matrices. La matriz A contiene el tablero que se está utilizando, la M contiene la base de datos de las jugadas (que se actualizan tras cada victoria o derrota), la W contiene los datos a partir de los cuales el programa reconoce una poten-

cial victoria suya o de su oponente y la D contiene las jugadas de la partida en curso, de modo que éstas puedan utilizarse para modificar la base de datos después de cada partida.

Como puede verse, desde la línea 1350, comienza con una base de datos que consiste meramente en los números 2, 6, 8, 4, 7, 3, 1, 9, 5 y 2. Se trata, como ya indicaba previamente, de una secuencia de jugadas particularmente mala, lo que asegura que perderá una proporción relativamente alta de sus primeras partidas. Si alguien lo dudara, no tiene más que poner esas jugadas en el tablero que usamos en este juego:

```
 1  :  2  :  3
---------------
 4  :  5  :  6
---------------
 7  :  8  :  9
```

Se ha de observar que el programa no mueve necesariamente en el orden expuesto. Lo intenta, pero puede encontrarse con que esa casilla en concreto está ocupada en ese momento. Asimismo, tampoco usa esta secuencia hasta haber verificado los datos preprogramados en relación con el bloqueo de su contrincante en el caso de que éste vaya a hacer tres en raya y, por otro lado, haber intentado completar su propio tres en raya.

El observar cómo el programa va aprendiendo es particularmente fascinante. Así pues, parte de éste nos informará de su proceso de aprendizaje al final de cada partida, mostrándonos la secuencia que está almacenando. La actualización de la base de datos, así como la tarea de informarnos sobre ella, la lleva a cabo la sección del programa comprendida entre las líneas 300 y 480:

```
300 REM ACTUALIZACION DE LA BASE DE DATOS
310 FOR B = 1 TO 5
320 FOR J = 2 TO 9
330 IF M(J) = D(B) THEN GOSUB 370
340 NEXT J
350 NEXT B
360 GOTO 430
370 REM ** REORDENACION DE LOS ELEMENTOS DE LA
         MATRIZ M **
380 TEMP = M(J + BANDERA)
390 M(J + BANDERA) = M(J)
400 M(J) = TEMP
410 J = 9
420 RETURN
430 PRINT: PRINT
440 PRINT "ESTA ES MI PRIORIDAD ACTUALIZADA"
450 PRINT: PRINT
460 FOR J = 1 TO 9
470 PRINT M(J); " ";
480 NEXT J
```

Lo que viene a continuación es el desarrollo de la base de datos de una versión en la que el contrincante era el poco inteligente generador de números aleatorios de mi propio ordenador. Pese a la falta de un oponente adecuado, el programa se las arregló para aprender con bastante rapidez. Se puede ver lo rápido que TICTAC descubre el valor de mover a la casilla central (la número cinco en nuestro tablero):

2	8	6	4	7	3	1	5	9
2	6	4	8	7	3	5	1	9
6	2	4	8	3	7	1	5	9
2	4	6	8	3	7	5	1	9
4	6	2	8	7	3	1	5	9
6	4	2	8	7	3	5	1	9
4	6	2	8	7	5	3	1	9
6	4	2	8	5	7	3	1	9
6	2	4	5	8	7	3	9	1
2	6	5	8	4	7	3	9	1
2	5	6	4	8	7	3	9	2
5	2	6	8	4	7	3	2	9
2	6	5	4	8	7	3	2	9
2	5	6	8	4	7	2	3	9
5	6	2	4	8	2	7	3	9
6	5	4	2	8	2	7	3	9
5	4	6	2	8	2	7	3	9
4	5	2	6	8	7	3	2	9
5	4	6	2	8	7	3	2	9
4	5	2	8	6	7	3	2	9
5	4	2	6	8	7	3	2	9
4	5	6	2	8	7	3	2	9
5	6	4	2	8	7	3	2	9

Seguidamente, utilicé la última secuencia de la ejecución automática (con la única diferencia que cambié uno de los doses repetidos por un uno), en vez de la secuencia de apertura que se da en el listado completo del programa y me puse yo mismo a jugar contra el programa, intentando ganarle en todas las partidas. Como puede verse a continuación, el programa continuaba aprendiendo:

4	5	6	2	8	7	3	1	2
4	5	6	2	8	7	3	1	2
4	6	5	2	8	3	7	1	2
6	5	4	2	8	3	7	1	2
5	4	6	2	8	3	7	1	2
5	4	6	2	8	3	7	1	2
5	4	6	2	8	3	7	1	2
4	5	6	2	8	3	7	1	2
5	4	2	6	8	3	7	1	2

El programa sufrió una ligera modificación al introducirle una nueva secuencia de apertura; una secuencia que a mi juicio era la mejor que podía darle. El ordenador jugó primero contra un jugador humano con el siguiente desarrollo (o falta del mismo) de su base de datos:

5	1	3	7	9	2	4	6	8
1	3	7	5	9	2	4	6	8
3	7	5	1	9	2	4	6	8
7	5	3	1	9	2	4	6	8
5	3	7	1	9	2	4	6	8
5	3	7	1	9	2	4	6	8

Posteriormente se le puso a jugar contra un contrincante que jugaba al azar. Puede verse que es muy poco lo que puede aprender de éste, dando la impresión de que simplemente estuviera bajarando unos cuantos números sin saber muy bien qué hacer con ellos.

1	5	3	9	7	2	4	6	8
5	1	9	3	7	2	4	6	8
1	9	5	3	7	2	4	6	8
1	9	5	3	7	2	4	6	8
9	5	1	3	7	2	4	6	8
5	1	9	3	7	2	4	6	8
1	9	5	3	7	2	4	6	8
1	5	9	3	7	2	4	6	8
5	9	1	3	7	2	4	6	8
5	9	1	3	7	2	4	6	8
9	1	5	3	7	2	4	6	8
1	5	9	3	7	2	4	6	8

Finalmente, volví a la secuencia de apertura que tenía al principio, dejando que el ordenador tomara la delantera al generador de números aleatorios. Tras 90 partidas, la secuencia fue la siguiente:

3	7	4	5	8	6	9	2	2
7	3	5	4	8	6	9	2	2
7	5	4	3	6	8	9	2	2
7	4	5	3	6	8	9	2	2
4	5	7	6	3	8	2	9	2
4	7	5	6	3	8	2	9	2
7	4	5	6	3	8	2	9	2
7	4	5	6	3	8	2	9	2
4	7	5	3	8	6	9	2	2
7	4	5	3	8	6	9	2	2
4	7	5	3	8	6	9	2	2
7	4	5	3	8	6	9	2	2
4	7	5	3	8	6	9	2	2
7	4	5	3	8	6	9	2	2
7	4	5	3	8	6	9	2	2

El programa tiene un punto flaco. Aunque de un modo u otro acaba por aprender, parece que se deja convencer con demasiada facilidad de la conveniencia de intercambiar los números, aun cuando esto no siempre le ayude a jugar mejor. Puede que haya quien desee seguir investigando sobre el modo en que el ordenador utiliza lo que aprende en cada partida.

Como ya he indicado antes, la estrategia de juego de TICTAC no proviene tan sólo de su base de datos. Tiene también información de los tres en raya que está tratando de completar (y de los que está tratando de impedir que complete su contrincante). Esta es la sección del código que busca la jugada antes de utilizar la base de datos:

```
540 REM MOVIMIENTO DEL ORDENADOR
550 P = ASC("O")
560 X = 0
570 J = 1
580 IF A(W(J)) = A(W(J + 1)) AND A(W(J + 2)) = 32
    AND A(W(J)) = P THEN X = W(J + 2): GOTO 750
590 IF A(W(J)) = A(W(J + 2)) AND A(W(J + 1)) = 32
    AND A(W(J)) = P THEN X = W(J + 1): GOTO 750
600 IF A(W(J + 1)) = A(W(J + 2)) AND A(W(J)) = 32
    AND A(W(J + 1)) = P THEN X = W(J): GOTO 750
610 IF J < 21 THEN J = J + 3: GOTO 580
620 IF P = ASC("O") THEN P = ASC("X"): GOTO 570
```

En primer lugar, trata de encontrar una jugada que le permita ganar (cuando P es igual al código ASCII de la letra "O") y luego intenta encontrar una jugada que bloquee al contrincante (con P ajustado igual al código de la ficha del contrincante, la "X"). Si no consigue encontrar una jugada en esta sección, hace entrar los datos de la base de datos:

```
630 REM ** SI NO SE GANA SE BUSCA UN MOVIMIENTO
        DE BLOQUEO **
640 REM * ENTONCES SE USA LA SIGUIENTE SECCION *
650 J = 1
660 IF A(M(J)) = 32 THEN X = M(J): GOTO 750
670 IF J < 10 THEN J = J + 1: GOTO 660
```

Si en ésta tampoco encuentra una jugada, lo intenta con números al azar:

```
680 H = 0
690 H = H + 1
700 X = INT(RND(1) * 9): IF A(X) = 32 THEN 750
710 IF H < 100 THEN 690
720 R$ = "D": REM ES SIMPLEMENTE UN DIBUJO
```

Cuando ya ha encontrado la jugada, mueve su ficha y luego se asegura de que, si todas las posiciones están ocupadas y no se le ha asignado ningún valor a R$ (que representa al "resultado" y que está ajustado a "G" en el caso de una victoria, "P" en el caso de una derrota y "D" en el de un empate), la partida debe de haber quedado empatada.

```
750 REM REALIZA EL MOVIMIENTO
760 A(X) = ASC("O")
770 CCONTADOR = CCONTADOR + 1
780 D(CCONTADOR) = X
790 BANDERA = 0
800 FOR J = 1 TO 9
810 IF A(J) = 32 THEN BANDERA = 1
820 NEXT J
830 IF BANDERA = 0 AND R$="" THEN R$ ="D"
840 REM SI TODAS LAS CASILLAS ESTAN LLENAS Y R$
         ESTA VACIO, ENTONCES ES SIMPLEMENTE UN
         DIBUJO
850 RETURN
```

Después de cada jugada, humana o de la máquina, aparece la rutina de COMPROBACION DE VICTORIA:

```
870 REM COMPRUEBA LA VICTORIA
880 J = 1
890 IF A(W(J)) = 32 THEN J = J + 3
900 IF J > 23 THEN RETURN
910 IF A(W(J)) = A(W(J + 1)) AND A(W(J)) = A(W(J
    + 2)) THEN 940
920 IF J < 22 THEN J = J + 3: GOTO 890
930 RETURN
940 IF A(W(J)) = ASC("O") THEN R$ = "G": REM EL
    ORDENADOR GANA
950 IF A(W(J)) = ASC("X") THEN R$ = "P": REM EL
    ORDENADOR PIERDE
960 RETURN
```

Este es el listado completo del programa TICTAC, para quien quisiera investigar por su cuenta un tema tan interesante como la educación de las máquinas (para usuarios del ZX Spectrum, véase Apéndices):

```
10 REM TICTAC
20 GOSUB 1180: REM INICIALIZACION
30 REM *** REQUISITOS PREVIOS AL JUEGO ***
40 FOR J = 1 TO 9
```

```
 50 A(J) = 32
 60 NEXT J
 70 FOR J = 1 TO 5
 80 D(J) = 0
 90 NEXT J
100 CCONTADOR = 0
110 R$ = ""
120 GOSUB 1070: REM IMPRESION DEL TABLERO
130 REM ** CICLO PRINCIPAL **
140 GOSUB 540: REM MOVIMIENTO DEL ORDENADOR
150 GOSUB 1070: REM IMPRESION DEL TABLERO
160 GOSUB 870: REM COMPRUEBA LA VICTORIA
170 IF R$ <> "" THEN 240
180 GOSUB 980: REM SE ACEPTA EL MOVIMIENTO DE LA
    PERSONA
190 GOSUB 1070: REM IMPRESION DEL TABLERO
200 GOSUB 870: REM COMPRUEBA LA VICTORIA
210 IF R$ = "" THEN 140
220 REM ** FIN DEL CICLO PRINCIPAL **
230 REM ************
240 REM FIN DEL JUEGO
250 GOSUB 1070: REM IMPRESION DEL TABLERO
260 PRINT: PRINT
270 IF R$ = "G" THEN PRINT TAB(8); "YO GANO":
    BANDERA = -1
280 IF R$ = "P" THEN PRINT TAB(8); "TU GANAS":
    BANDERA = 1
290 IF R$ = "D" THEN PRINT TAB(6); "ES SIMPLEMEN
    TE UN DIBUJO": GOTO 430
300 REM ACTUALIZACION DE LA BASE DE DATOS
310 FOR B = 1 TO 5
320 FOR J = 2 TO 9
330 IF M(J) = D(B) THEN GOSUB 370
340 NEXT J
350 NEXT B
360 GOTO 430
370 REM ** REORDENACION DE LOS ELEMENTOS DE LA
        MATRIZ M **
380 TEMP = M(J + BANDERA)
390 M(J + BANDERA) = M(J)
400 M(J) = TEMP
410 J = 9
420 RETURN
430 PRINT: PRINT
440 PRINT "ESTA ES MI PRIORIDAD ACTUALIZADA"
450 PRINT: PRINT
460 FOR J = 1 TO 9
470 PRINT M(J); " ";
480 NEXT J
490 PRINT: PRINT
500 PRINT "PULSE LA TECLA 'RETURN' PARA CONTINUAR
    "
510 INPUT A$
520 GOTO 30
530 REM ***********************
540 REM MOVIMIENTO DEL ORDENADOR
550 P = ASC("O")
```

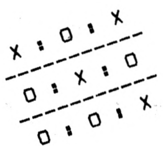
```
560 X = 0
570 J = 1
580 IF A(W(J)) = A(W(J + 1)) AND A(W(J + 2)) = 32
    AND A(W(J)) = P THEN X = W(J + 2): GOTO 750
590 IF A(W(J)) = A(W(J + 2)) AND A(W(J + 1)) = 32
    AND A(W(J)) = P THEN X = W(J + 1): GOTO 750
600 IF A(W(J + 1)) = A(W(J + 2)) AND A(W(J)) = 32
    AND A(W(J + 1)) = P THEN X = W(J): GOTO 750
610 IF J < 21 THEN J = J + 3: GOTO 580
620 IF P = ASC("O") THEN P = ASC("X"): GOTO 570
630 REM ** SI NO SE GANA SE BUSCA UN MOVIMIENTO
        DE BLOQUEO **
640 REM * ENTONCES SE USA LA SIGUIENTE SECCION *
650 J = 1
660 IF A(M(J)) = 32 THEN X = M(J): GOTO 750
670 IF J < 10 THEN J = J + 1: GOTO 660
680 H = 0
690 H = H + 1
700 X = INT(RND(1) * 9): IF A(X) = 32 THEN 750
710 IF H < 100 THEN 690
720 R$ = "D": REM ES SIMPLEMENTE UN DIBUJO
730 RETURN
740 REM ********************
750 REM REALIZA EL MOVIMIENTO
760 A(X) = ASC("O")
770 CCONTADOR = CCONTADOR + 1
780 D(CCONTADOR) = X
790 BANDERA = 0
800 FOR J = 1 TO 9
810 IF A(J) = 32 THEN BANDERA = 1
820 NEXT J
830 IF BANDERA = 0 AND R$="" THEN R$ ="D"
840 REM SI TODAS LAS CASILLAS ESTAN LLENAS Y R$
        ESTA VACIO, ENTONCES ES SIMPLEMENTE UN
        DIBUJO
850 RETURN
860 REM ********************
870 REM COMPRUEBA LA VICTORIA
880 J = 1
890 IF A(W(J)) = 32 THEN J = J + 3
900 IF J > 23 THEN RETURN
910 IF A(W(J)) = A(W(J + 1)) AND A(W(J)) = A(W(J
    + 2)) THEN 940
920 IF J < 22 THEN J = J + 3: GOTO 890
930 RETURN
940 IF A(W(J)) = ASC("O") THEN R$ = "G": REM EL
    ORDENADOR GANA
950 IF A(W(J)) = ASC("X") THEN R$ = "P": REM EL
    ORDENADOR PIERDE
960 RETURN
970 REM ***********************
980 REM MOVIMIENTO DE LA PERSONA
990 PRINT: PRINT
1000 PRINT "REALICE SU MOVIMIENTO"
1010 INPUT "MOVIMIENTO"; MOVIMIENTO
1020 IF MOVIMIENTO < 1 OR MOVIMIENTO > 9 THEN 1010
1030 IF A(MOVIMIENTO) <> 32 THEN 1010
```

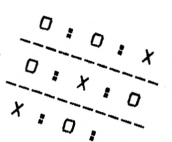
```
1040 A(MOVIMIENTO) = ASC("X")
1050 RETURN
1060 REM *********************
1070 REM IMPRESION DEL TABLERO
1080 CLS
1090 PRINT: PRINT: PRINT
1100 PRINT " 1 : 2 : 3          "; CHR$(A(1)); " : ";
              CHR$(A(2)); " : "; CHR$(A(3))
1110 PRINT "----------      -----------"
1120 PRINT " 4 : 5 : 6          "; CHR$(A(4)); " : ";
              CHR$(A(5)); " : "; CHR$(A(6))
1130 PRINT "----------      -----------"
1140 PRINT " 7 : 8 : 9          "; CHR$(A(7)); " : ";
              CHR$(A(8)); " : "; CHR$(A(9))
1150 PRINT
1160 RETURN
1170 REM **************
1180 REM INICIALIZACION
1190 CLS
1200 DIM A(9): REM TABLERO
1210 DIM M(10): REM ACCESO A LA BASE DE DATOS
1220 DIM W(24): REM DATOS DE VICTORIA O BLOQUEO
1230 DIM D(5): REM ACCESO AL MOVIMIENTO EN EL
     JUEGO ACTUAL
1240 REM DATOS DE VICTORIA O BLOQUEO
1250 FOR J = 1 TO 24
1260 READ W(J)
1270 NEXT J
1280 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
1290 DATA 1, 4, 7, 2, 5, 8, 3, 6, 9
1300 DATA 1, 5, 9, 3, 5, 7
1310 REM BASE INICIAL DE DATOS
1320 FOR J = 1 TO 10
1330 READ M(J)
1340 NEXT J
1350 DATA 2, 6, 8, 4, 7, 3, 1, 9, 5, 2
1360 RETURN
```

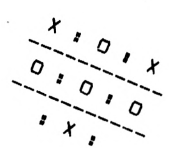
Si el lector deseara experimentar con un contrincante automático que jugara al azar, puede que le interese utilizar éste, que es el que yo mismo usé para este capítulo del libro:

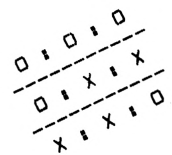
```
4500 REM MOVIMIENTO ALEATORIO DE UNA PERSONA
4510 P = 0
4520 P = P + 1
4530 MOVIMIENTO = INT(RND(1) * 9 + 1)
4540 IF A(MOVIMIENTO) = 32 THEN A(MOVIMIENTO) =
     ASC("X"):RETURN
4550 IF P < 100 THEN 4520
4560 R$ = "D"
4570 RETURN
```

Para que se dispare este contrincante, que aunque incansable, carece totalmente de inteligencia, basta con sustituir la línea 140 por GOSUB 4500.











3
Un programa que razona

De un programa que aprende pasamos a SILOGISMOS, un programa que razona. Dadas dos sentencias relacionadas, SILOGISMOS es capaz de deducir una tercera que contiene una información que no se le había dado explícitamente.

El programa funciona con silogismos. Un silogismo es un tipo de razonamiento deductivo. Aristóteles elaboró las reglas que determinan su validez. Generalmente, tiene la siguiente forma:

A es un B
C es un A
Luego, C es un B

Las primeras dos líneas de un silogismo se denominan premisas o proposiciones, la tercera línea es la conclusión.

Un perro es un animal
Un animal es peludo
Luego, un perro es peludo

Antes de comentar el programa y su origen en detalle, veremos cómo funciona. Ignore todo el material que aparece entre paréntesis antes de la conclusión, ya que se ha incluido para que el lector vea su funcionamiento real. Comprenderá de qué se trata cuando llegue a la explicación del programa.

El signo recordatorio '?' aparece cuando SILOGISMOS está esperando una entrada

(input). '< DE ACUERDO' aparece cuando el programa ha entendido y aceptado esa entrada.

```
? UN AGUILA ES UN PAJARO
      >  DE ACUERDO

? UN PAJARO ES UN SER CON ALAS
      >  DE ACUERDO

? ES UN AGUILA UN SER CON ALAS
          (ESTOY BUSCANDO 'AGUILA')
          ( ENCONTRADA EN 1 1 )
      > SI

?
```

A medida que se va ejecutando, el programa elabora una base de datos con las proposiciones, a la que puede remitirse en cualquier momento dentro de esa misma ejecución. Seguidamente probé con este otro par de proposiciones:

```
? UN PAJARO ES UN AVE VOLADORA
      >  DE ACUERDO

? ES UN AGUILA UN AVE VOLADORA
         (ESTOY BUSCANDO 'AGUILA')
          ( ENCONTRADA EN 1 1 )
      > SI

? ES UN PAJARO UN SER CON ALAS
         (ESTOY BUSCANDO 'PAJARO')
          (ENCONTRADA EN 1 2 )
      > SI

?
```

SILOGISMOS aceptará, para añadirla a su base de datos, cualquier sentencia con la forma siguiente:

Un … es un …

Esta sentencia puede incluir 'el', 'la', 'un', 'una', ya que el programa incluye un análisis gramatical del lenguaje, lo que le capacita para entenderlos y utilizarlos correctamente. Por consiguiente, todos ellos son válidos, aunque se ha de tener en cuenta que el programa no sabrá qué hacer si se encuentra con un 'el', por ejemplo, situado tras un 'es' en mitad de frase:

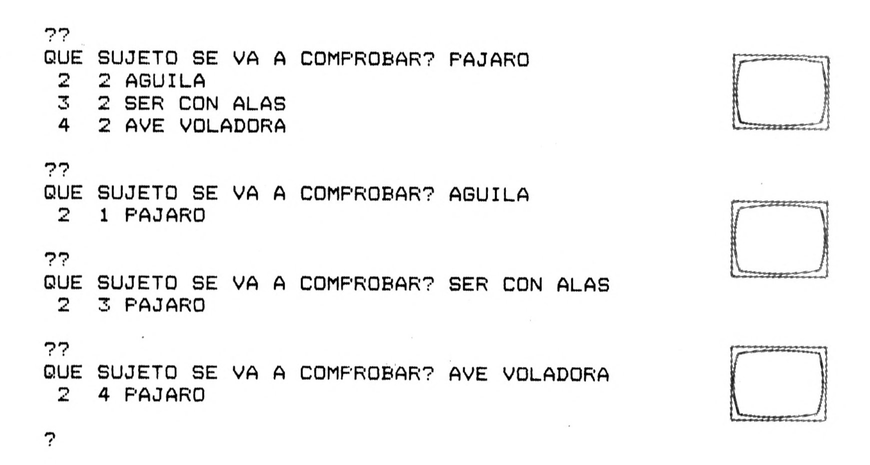
Un … es un …
El … es una …
La … es un …

El programa se lanza a la "carrera deductiva" cuando se encuentra con una frase que empieza por 'es':

Es ... un ...
Es un ... un ...

Si uno se limita a presionar la tecla RETURN, sin introducir ninguna información *(input)*, el programa dará por terminada la ejecución (aunque puede volverse a empezar, sin que pierda los datos que ha acumulado, mediante el GOTO 30).

En el caso de introducir un signo de interrogación cuando aparece el signo recordatorio (es decir, tras la primera interrogación con la que SILOGISMOS nos recuerda que está esperando una entrada), podremos descubrir lo que SILOGISMOS tiene acumulado en la memoria, todo ello ordenado en las categorías, cada cual con su encabezamiento respectivo, que SILOGISMOS ha creado a tal efecto. Una vez introducida la interrogación, el programa preguntará "¿QUE SUJETO SE VA A COMPROBAR?". En este punto se introduce el encabezamiento que se desea que investigue el programa:



```
??
QUE SUJETO SE VA A COMPROBAR? PAJARO
 2  2 AGUILA
 3  2 SER CON ALAS
 4  2 AVE VOLADORA

??
QUE SUJETO SE VA A COMPROBAR? AGUILA
 2  1 PAJARO

??
QUE SUJETO SE VA A COMPROBAR? SER CON ALAS
 2  3 PAJARO

??
QUE SUJETO SE VA A COMPROBAR? AVE VOLADORA
 2  4 PAJARO

?
```

SILOGISMOS sacará con frecuencia conclusiones sorprendentes; unas conclusiones que hacen caso omiso de toda la evidencia que podamos tener (y hablo por mí en este caso):

```
?ISABEL ES UNA TONTA
       > DE ACUERDO

?UNA TONTA ES UNA IDIOTA
       > DE ACUERDO

?ES ISABEL UNA IDIOTA
          (ESTOY BUSCANDO 'ISABEL')
           ( ENCONTRADA EN 1 1 )
      > SI

?
```

```
??
QUE SUJETO SE VA A COMPROBAR? ISABEL
 2  1 TONTA

??
QUE SUJETO SE VA A COMPROBAR? TONTA
 2  2 ISABEL
 3  2 IDIOTA

??
QUE SUJETO SE VA A COMPROBAR? IDIOTA
2  3 TONTA

?
```

Aunque SILOGISMOS puede engañarse a veces y sacar conclusiones totalmente absurdas, por lo general su razonamiento es bastante consistente:

```
? UN AVESTRUZ ES UNA IDIOTA
      >  DE ACUERDO

? ES ISABEL UN AVESTRUZ
         (ESTOY BUSCANDO 'ISABEL')
          ( ENCONTRADA EN 1 1 )
      > NO

? ES UN AVESTRUZ UNA TONTA
         (ESTOY BUSCANDO 'AVESTRUZ')
          ( ENCONTRADA EN 1 6 )
      > SI

?
```

SILOGISMOS funciona con una matriz bidimensional de cadenas alfanuméricas, Z$, a la que se remiten, a modo de referencias cruzadas, todas las proposiciones que se introducen y no sacando conclusiones más que a partir de estas referencias.

Este mecanismo no es muy difícil de comprender si uno intenta visualizar lo que sucede al introducir las sentencias. Cuando tecleamos una sentencia como ISABEL ES UNA TONTA, el programa ignora ES UNA, utilizando ISABEL como encabezamiento de un nuevo apartado o categoría en su base de datos y metiendo TONTA bajo este apartado. Una segunda sentencia del tipo UN TONTO ES UN IDIOTA permite que el programa abra un segundo nuevo apartado en su archivo, que encabezará con TONTO y en el que incluirá IDIOTA. Así pues, cuando le preguntemos al programa ES ISABEL UNA IDIOTA, éste mirará primero si tiene en su archivo una categoría cuyo encabezamiento sea ISABEL. Al descubrir que sí cuenta con dicha categoría, mirará cuál es el primer sujeto incluido en ella y se encontrará con TONTA.

Acto seguido mira si tiene una categoría llamada TONTO. Cuando ve que así es, revisa todos los sujetos archivados bajo este encabezamiento y descubre el sujeto ISABEL. Gracias a estas referencias cruzadas, sabe que la respuesta a la pregunta ES ISABEL UNA IDIOTA es sí.

Este es el procedimiento que SILOGISMOS sigue siempre, independientemente de cuáles sean las series de sentencias que uno introduce en el programa. Dado que en una matriz de 25 × 25, como la que utilizamos con este programa, hay mucho espacio, quizá haya quien desee guardar las bases de datos de algunos sujetos.

Obviamente, ambas series, ISABEL ES UNA IDIOTA y EL AGUILA ES UN PAJARO, se trataron por separado. El gráfico que aparece a continuación muestra el orden de almacenamiento interno, es decir, la manera en que están archivados los datos en la serie EL AGUILA ES UN PAJARO. Puede que esto nos facilite la tarea de comprender cómo SILOGISMOS archiva, para posteriormente acceder a ellas, las proposiciones a partir de las cuales saca conclusiones:

	1	2	3	4
1	AGUILA	PAJARO	SER CON ALAS	AVE VOLADORA
2	PAJARO	AGUILA	PAJARO	PAJARO
3		SER CON ALAS		
4		AVE VOLADORA		
5				

Cuando el programa se encuentra con un nuevo sujeto (y éste es el primer nombre de la proposición), atraviesa la parte superior de la matriz buscando por orden primero en el 1,1, después en el 1,2, luego en el 1,3 y así sucesivamente hasta que encuentra un espacio sin usar. Esto es lo que sucede cuando uno introduce una proposición como EL AGUILA ES UN PAJARO al principio de la ejecución. El 1,1 está libre, de modo que el programa almacena AGUILA en éste y PAJARO bajo aquél en el 2,1.

Tras esto, invierte el orden de los dos nombres y abre una categoría llamada PAJARO que coloca en el 1,2, archivando en ésta AGUILA (en el 2,2). Cuando se le introduzca otra sentencia que se refiera a un sujeto para el que el programa ya tiene establecida una categoría, como UN PAJARO ES UN SER CON ALAS, almacenará o archivará la información SER CON ALAS en el 3,2 y abrirá otra categoría con el encabezamiento de SER CON ALAS en el 3,1, bajo el cual archivará PAJARO.

Y así continúa, archivando mediante referencias cruzadas toda la información que recibe con el fin de poder acceder a ella posteriormente. La última sentencia que introdujimos en esta ejecución fue UN PAJARO ES UN AVE VOLADORA, y como era de esperar SILOGISMOS archivó AVE VOLADORA en el primer espacio vacío que encontró bajo el encabezamiento PAJARO (esto es, en el 4,2) y abrió una nueva categoría llamada AVE VOLADORA en el 1,4, bajo la que archivó PAJARO.

Cuando se introduce un signo de interrogación para comprobar el contenido de un archivo, el ordenador va recorriendo por orden la hilera de encabezamientos (es decir, empieza en el 1,1, pasa al 1,2, de ahí al 1,3 y así sucesivamente) hasta que encuentra el sujeto que busca. Si llega al final (esto es, al 1,25) sin haberlo encontrado, nos hará saber que no tiene datos archivados sobre ese sujeto. Cuando encuentra el sujeto (como PAJARO en el 1,2), procede a imprimir los contenidos archivados bajo ese encabezamiento. En este caso serían, pues, AGUILA, SER CON ALAS, AVE VOLADORA.

Cuando llega el momento de tomar una decisión sobre si ES UN AGUILA UN AVE

VOLADORA (el mecanismo de decisiones se dispara cuando la entrada [*input*] del usuario comienza por la palabra OR), el programa comprueba en primer lugar, recorriendo para ello la hilera superior, si tiene información almacenada sobre el primer nombre de la pregunta. Si descubre que tiene, lo hace saber al usuario (ESTOY BUSCANDO 'AGUILA' ENCONTRADA EN 1,1) y después pasa revista de arriba a abajo a los nombres archivados bajo este encabezamiento, descubriendo PAJARO en el 2,1. Vuelve entonces a la primera hilera para buscar AVE VOLADORA. Tras encontrarlo en 1,4, revisa los nombres agrupados en esta categoría y encuentra PAJARO (en el 2,4). Llegado a este punto y habiendo encontrado una relación entre las dos palabras que ocupan su pensamiento en ese momento (AGUILA y AVE VOLADORA), puede concluir que la respuesta a la pregunta ES UN AGUILA UN AVE VOLADORA es, sin duda, SI. SILOGISMOS nos dirá entonces cuál es su conclusión.

El programa de SILOGISMOS

Aquí tenemos el principio del listado de SILOGISMOS en donde el programa procesa la entrada del usuario. La línea 40 envía la acción a la 960 en el caso de haber introducido una pregunta marcada.

```
 10 REM SILOGISMOS
 20 GOSUB 1100: REM INICIALIZACION
 30 PRINT: INPUT A$
 40 IF A$ = "?" THEN 960
 50 IF A$ = "" THEN PRINT "FIN DEL PROGRAMA": END
 60 BANDERA = 0
 70 REM ATENCION! HAY UN ESPACIO ANTES DE LAS
         COMILLAS EN LAS PROXIMAS 5 LINEAS
 80 IF LEFT$(A$, 3) = "ES " THEN 510: REM CONCLU
    SIONES
 90 IF LEFT$(A$, 3) = "EL " THEN A$ = MID$(A$, 4)
100 IF LEFT$(A$, 3) = "LA " THEN A$ = MID$(A$, 4)
110 IF LEFT$(A$, 3) = "UN " THEN A$ = MID$(A$, 4)
120 IF LEFT$(A$, 4) = "UNA " THEN A$= MID$(A$, 5)
130 X = LEN(A$)
140 N = 0
150 N = N + 1
160 IF MID$(A$, N, 1) = " " THEN B$ = LEFT$( A$,
    N - 1): GOTO 190: REM EXTRAE EL PRIMER NOMBRE
170 IF N < X THEN 150
180 PRINT "NO COMPRENDO": GOTO 30
190 K = 4
200 IF MID$(A$, N + 1, 1) = "F" THEN K = 5
210 C$ = MID$(A$, N + K): REM PREPARATIVOS DE LA
    FRASE
220 REM ** ELIMINACION DE ARTICULOS **
230 IF LEFT$(C$, 4)= "UNA " THEN C$ = MID$(C$, 5)
240 IF LEFT$(C$, 3) = "UN " THEN C$ = MID$(C$, 4)
250 IF LEFT$(C$, 3) = "EL " THEN C$ = MID$(C$, 4)
260 IF LEFT$(C$, 3) = "LA " THEN C$ = MID$(C$, 4)
```

La línea 80 detecta el 'ES' al principio de la entrada, lo que indica que el usuario pide a SILOGISMOS que intente llegar a una conclusión. Esta envía la acción a la 510, en donde comienza la rutina de conclusión.

Las líneas 90 a 120 quitan EL, LA, UN y UNA del inicio de la entrada, de la sentencia, de modo que A$ empezará entonces con el nombre que se utilizará para encabezar un archivo o categoría.

La siguiente rutina, de las líneas 130 a 260, separa la entrada en dos palabras; las líneas 130 a 170 se ocupan del primer nombre y dispararán NO COMPRENDO (desde la línea 180) si la entrada no responde al formato específico. Las líneas comprendidas entre la 190 y la 260 extraen la segunda palabra. La línea 200 comprueba si la frase que queda, tras haber quitado el primer nombre, empieza por "F", en cuyo caso asume que la palabra central es "FUE". Esto permite que el programa acepte tanto frases de este tipo:

DAVID FUE UN GRAN HOMBRE

como de éste otro:

ISABEL ES UNA TONTA

Una vez extraídas las palabras importantes (y habiendo fijado B$ para la primera y C$ para la segunda) el programa procede a archivarlas en su base de datos. Se ha de recordar que esta sección del código se usa sólo para "guardar" la información. Posteriormente la retomará la sección del programa "encargada de llegar a una conclusión".

Seguidamente el programa repasará la parte superior de su fichero (los encabezamientos) para comprobar: *a*) si ya cuenta con un fichero sobre ese sujeto y, de no ser así, *b*) si le queda espacio para abrir uno nuevo. Si no le queda espacio, imprimirá en la línea 340 el siguiente mensaje: NO ME QUEDA ESPACIO PARA ALMACENAR MAS SUJETOS.

```
270 REM ** ALMACENAMIENTO DE LA INFORMACION **
280 REM ** PRIMERO SE COMPRUEBA SI SE PUEDE ENCON
        TRAR UN SUJETO, ANTES DE UN ESPACIO EN
        BLANCO **
290 N = 0
300 N = N + 1
310 IF Z$(1, N) = B$ THEN 350: REM EXISTE UN SUJE
    TO AL PRINCIPIO DE LA FRASE
320 IF Z$(1, N) = "" THEN Z$(1, N) = B$: GOTO 350
330 IF N < 25 THEN 300
340 PRINT "NO ME QUEDA ESPACIO PARA ALMACENAR
           MAS SUJETOS"
```

A la siguiente rutina, a partir de la línea 350, se llega una vez que el programa ha descubierto que ya tiene un fichero sobre ese sujeto (línea 310), o bien ha encontrado espacio para crear uno nuevo, lo que, en realidad, hace nada más encontrarlo (línea 320).

```
350 REM ** EL PROGRAMA HA LLEGADO AQUI CON EL
         SUJETO ENCABEZANDO EL ALMACENAMIENTO **
360 REM ** AHORA SE PONE EL OBJETO DIRECTO DEBAJO
         DE EL **
370 K = 0
380 K = K + 1
390 IF Z$(K, N) = C$ THEN 430: REM LA INFORMACION
    YA HA SIDO ALMACENADA DEBAJO DEL ENCABEZAMIEN
    TO
400 IF Z$(K, N) = "" THEN Z$(K, N) = C$: GOTO 430
410 IF K < 25 THEN 380
420 PRINT "NO TENGO MAS ESPACIO PARA ALMACENAR
            OBJETOS DIRECTOS"
430 IF BANDERA = 1 THEN PRINT TAB(6); ">  DE
    ACUERDO": GOTO 30: REM SE HA REALIZADO EL
    INTERCAMBIO
440 REM ** AHORA SE INTERCAMBIAN EL SUJETO Y EL
         OBJETO Y SE ALMACENA DE NUEVO **
450 BANDERA = 1
460 M$ = B$
470 B$ = C$
480 C$ = M$
490 GOTO 280
```

SILOGISMOS no necesita archivar o almacenar AGUILA bajo el encabezamiento PAJARO más de una vez, aun cuando la línea UN AGUILA ES UN PAJARO haya sido introducida en el programa más de una vez. La línea 390 se encarga de que no se guarden las definiciones duplicadas. Una vez archivado el ‘‘objeto’’, el ordenador intercambia sujeto y objeto (líneas 450 a 480) y los archiva a la inversa. Es decir, si antes había reservado AGUILA como encabezamiento de un fichero o categoría en el que había incluido PAJARO, ahora reserva PAJARO como encabezamiento e incluye AGUILA como un elemento más de ese fichero.

Llegamos ahora a la parte verdaderamente interesante (al menos en términos de la ejecución que realiza SILOGISMOS), la sección en la que el programa llega a una conclusión a partir de las proposiciones que ha almacenado:

```
510 REM ** CONCLUSIONES **
520 REM ** PRIMERAMENTE SE FRAGMENTA LA FRASE **
530 A$ = MID$(A$, 4): REM SE IGNORA 'ES'
540 IF LEFT$(A$, 4) = "UNA " THEN A$ = MID$(A$,
    5): REM SE IGNORA 'UNA' CASO DE HABERLA
550 IF LEFT$(A$, 3) = "UN " OR LEFT$(A$, 3) =
    "EL " OR LEFT$(A$,3) = "LA " THEN A$ = MID$(A
    $, 4): REM SE IGNORA 'UN', 'EL' O 'LA' CASO
    DE HABERLO
560 REM ** SE COGE LA PRIMERA PALABRA - F$ **
570 X = LEN(A$)
```

```
580 N = 0
590 N = N + 1
600 IF MID$(A$, N, 1) = " " THEN F$ = LEFT$(A$,
    N - 1): GOTO 630
610 IF N < X THEN 590
620 PRINT TAB(6); "> NO COMPRENDO": GOTO 30
```

En primer lugar se suprime de la entrada *(input)* el ES inicial junto con UNA (línea 540), UN, EL, o LA (línea 550) en el caso de que aparezcan (esto significa que podrá entendérselas tanto con ES UN AGUILA UN PAJARO como con ES ISABEL UNA IDIOTA). Esta sección del código se encarga de la primera palabra fijándola igual a F$. La siguiente sección extrae la segunda palabra y la fija igual a S$.

```
630 REM ** AHORA SE COGE LA SEGUNDA PALABRA - S$
        **
640 S$ = MID$(A$, N)
650 IF LEFT$(S$,4) = " UN " THEN S$ = MID$(A$,
    N + 4): GOTO 680
660 IF LEFT$(S$,5) = " UNA " THEN S$ = MID$(A$,
    N + 5): GOTO 680
670 S$=MID$(A$, N + 1)
680 PRINT TAB(9);"(ESTOY BUSCANDO '"; F$; "')"
690 X = 0
700 X = X + 1
710 IF Z$(1, X) = F$ THEN PRINT TAB(10); "( ENCON
    TRADA EN 1"; X; ")": GOTO 750
720 IF X < 25 THEN 700
730 PRINT TAB(6); "NO PUEDO ENCONTRAR EL SUJETO":
    PRINT TAB(8); F$
740 GOTO 30
```

El programa nos hará saber qué es lo que está buscando (imprimiendo ESTOY BUSCANDO la "primera palabra", en la línea 680) y, si lo encuentra, nos dirá en qué lugar del archivo la ha localizado (imprimiendo ENCONTRADA EN..., en la línea 710). Si no puede encontrar la segunda palabra nos lo dirá en la línea 730 volviendo después al programa principal. Se disparará esta línea en el caso, por ejemplo, de que le preguntemos al programa algo así como ES ISABEL UN GENIO sin que éste se haya encontrado previamente con la palabra GENIO.

```
750 Y = 1
760 Y = Y + 1
770 IF Z$(Y, X) = S$ THEN PRINT TAB(6); "> SI":
    GOTO 30
780 IF Y < 25 THEN 760
```

```
790 Y = 1
800 Y = Y + 1
810 P$ = Z$(Y, X)
820 M = 0
830 M = M + 1
840 IF Z$(1, M) = P$ THEN 880
850 IF M < 25 THEN 830
860 IF Y < 25 THEN 800
870 PRINT TAB(6); "> NO": GOTO 30
880 Q = 1
890 Q = Q + 1
900 IF Z$(Q, M) = S$ THEN PRINT TAB(6); "> SI":
    GOTO 30
910 IF Q < 25 THEN 890
920 IF M < 25 THEN 830
930 GOTO 870
```

La siguiente sección del código llega a conclusiones. El primer trozo, de las líneas 750 a 780, dice SI en el caso de que la pregunta que le hicimos tuviera exactamente la misma forma que la proposición o frase con que originalmente le habíamos dado la información. Es decir, si le preguntamos ES UNA AGUILA UN PAJARO y previamente la habíamos dicho de modo explícito UN AGUILA ES UN PAJARO, esta primera parte se dará cuenta y nos dirá SI.

La siguiente sección, de las líneas 760 a 930, busca la palabra siguiendo el método que hemos visto antes y concluye ya sea SI (en la línea 900) o NO (en la 870).

```
 960 REM COMPRUEBA EL CONTENIDO DE UN ARCHIVO PAR
         TICULAR
 970 INPUT "QUE SUJETO SE VA A COMPROBAR"; H$
 980 T = 0
 990 T = T + 1
1000 IF Z$(1, T) = H$ THEN 1040
1010 IF T < 25 THEN 990
1020 PRINT "NO TENGO DATOS ALMACENADOS EN "; H$
1030 GOTO 30
1040 K = 1
1050 K = K + 1
1060 IF Z$(K, T) <> "" THEN PRINT K; T; Z$(K, T)
1070 IF K < 25 THEN 1050
1080 GOTO 30
```

Esta parte final es la que nos dice lo que el programa ha archivado bajo cada encabezamiento.

Lo que viene a continuación es el listado completo de SILOGISMOS: conque ya podemos disponernos a sacar nuestras propias conclusiones (para usuarios del ZX Spectrum, véase Apéndices).

```
 10 REM SILOGISMOS
 20 GOSUB 1100: REM INICIALIZACION
 30 PRINT: INPUT A$
 40 IF A$ = "?" THEN 960
 50 IF A$ = "" THEN PRINT "FIN DEL PROGRAMA": END
 60 BANDERA = 0
 70 REM ATENCION! HAY UN ESPACIO ANTES DE LAS
         COMILLAS EN LAS PROXIMAS 5 LINEAS
 80 IF LEFT$(A$, 3) = "ES " THEN 510: REM CONCLU
    SIONES
 90 IF LEFT$(A$, 3) = "EL " THEN A$ = MID$(A$, 4)
100 IF LEFT$(A$, 3) = "LA " THEN A$ = MID$(A$, 4)
110 IF LEFT$(A$, 3) = "UN " THEN A$ = MID$(A$, 4)
120 IF LEFT$(A$, 4) = "UNA " THEN A$= MID$(A$, 5)
130 X = LEN(A$)
140 N = 0
150 N = N + 1
160 IF MID$(A$, N, 1) = " " THEN B$ = LEFT$( A$,
    N - 1): GOTO 190: REM EXTRAE EL PRIMER NOMBRE
170 IF N < X THEN 150
180 PRINT "NO COMPRENDO": GOTO 30
190 K = 4
200 IF MID$(A$, N + 1, 1) = "F" THEN K = 5
210 C$ = MID$(A$, N + K): REM PREPARATIVOS DE LA
    FRASE
220 REM ** ELIMINACION DE ARTICULOS **
230 IF LEFT$(C$, 4)= "UNA " THEN C$ = MID$(C$, 5)
240 IF LEFT$(C$, 3) = "UN " THEN C$ = MID$(C$, 4)
250 IF LEFT$(C$, 3) = "EL " THEN C$ = MID$(C$, 4)
260 IF LEFT$(C$, 3) = "LA " THEN C$ = MID$(C$, 4)
270 REM ** ALMACENAMIENTO DE LA INFORMACION **
280 REM ** PRIMERO SE COMPRUEBA SI SE PUEDE ENCON
        TRAR UN SUJETO, ANTES DE UN ESPACIO EN
        BLANCO **
290 N = 0
300 N = N + 1
310 IF Z$(1, N) = B$ THEN 350: REM EXISTE UN SUJE
    TO AL PRINCIPIO DE LA FRASE
320 IF Z$(1, N) = "" THEN Z$(1, N) = B$: GOTO 350
330 IF N < 25 THEN 300
340 PRINT "NO ME QUEDA ESPACIO PARA ALMACENAR
           MAS SUJETOS"
350 REM ** EL PROGRAMA HA LLEGADO AQUI CON EL
        SUJETO ENCABEZANDO EL ALMACENAMIENTO **
360 REM ** AHORA SE PONE EL OBJETO DIRECTO DEBAJO
        DE EL **
370 K = 0
380 K = K + 1
390 IF Z$(K, N) = C$ THEN 430: REM LA INFORMACION
    YA HA SIDO ALMACENADA DEBAJO DEL ENCABEZAMIEN
    TO
400 IF Z$(K, N) = "" THEN Z$(K, N) = C$: GOTO 430
410 IF K < 25 THEN 380
420 PRINT "NO TENGO MAS ESPACIO PARA ALMACENAR
           OBJETOS DIRECTOS"
```

```
430 IF BANDERA = 1 THEN PRINT TAB(6); ">  DE
    ACUERDO": GOTO 30: REM SE HA REALIZADO EL
    INTERCAMBIO
440 REM ** AHORA SE INTERCAMBIAN EL SUJETO Y EL
        OBJETO Y SE ALMACENA DE NUEVO **
450 BANDERA = 1
460 M$ = B$
470 B$ = C$
480 C$ = M$
490 GOTO 280
500 REM ******************
510 REM ** CONCLUSIONES **
520 REM ** PRIMERAMENTE SE FRAGMENTA LA FRASE **
530 A$ = MID$(A$, 4): REM SE IGNORA 'ES'
540 IF LEFT$(A$, 4) = "UNA " THEN A$ = MID$(A$,
    5): REM SE IGNORA 'UNA' CASO DE HABERLA
550 IF LEFT$(A$, 3) = "UN " OR LEFT$(A$, 3) =
    "EL " OR LEFT$(A$,3) = "LA " THEN A$ = MID$(A
    $, 4): REM SE IGNORA 'UN', 'EL' O 'LA' CASO
    DE HABERLO
560 REM ** SE COGE LA PRIMERA PALABRA - F$ **
570 X = LEN(A$)
580 N = 0
590 N = N + 1
600 IF MID$(A$, N, 1) = " " THEN F$ = LEFT$(A$,
    N - 1): GOTO 630
610 IF N < X THEN 590
620 PRINT TAB(6); "> NO COMPRENDO": GOTO 30
630 REM ** AHORA SE COGE LA SEGUNDA PALABRA - S$
        **
640 S$ = MID$(A$, N)
650 IF LEFT$(S$,4) = " UN " THEN S$ = MID$(A$,
    N + 4): GOTO 680
660 IF LEFT$(S$,5) = " UNA " THEN S$ = MID$(A$,
    N + 5): GOTO 680
670 S$=MID$(A$, N + 1)
680 PRINT TAB(9);"(ESTOY BUSCANDO '"; F$; "')"
690 X = 0
700 X = X + 1
710 IF Z$(1, X) = F$ THEN PRINT TAB(10); "( ENCON
    TRADA EN 1"; X; ")": GOTO 750
720 IF X < 25 THEN 700
730 PRINT TAB(6); "NO PUEDO ENCONTRAR EL SUJETO":
    PRINT TAB(8); F$
740 GOTO 30
750 Y = 1
760 Y = Y + 1
770 IF Z$(Y, X) = S$ THEN PRINT TAB(6); "> SI":
    GOTO 30
780 IF Y < 25 THEN 760
790 Y = 1
800 Y = Y + 1
810 P$ = Z$(Y, X)
820 M = 0
830 M = M + 1
840 IF Z$(1, M) = P$ THEN 880
850 IF M < 25 THEN 830
```

```
860 IF Y < 25 THEN 800
870 PRINT TAB(6); "> NO": GOTO 30
880 Q = 1
890 Q = Q + 1
900 IF Z$(Q, M) = S$ THEN PRINT TAB(6); "> SI":
    GOTO 30
910 IF Q < 25 THEN 890
920 IF M < 25 THEN 830
930 GOTO 870
940 REM *****************
950 REM ****************************************
960 REM COMPRUEBA EL CONTENIDO DE UN ARCHIVO PAR
        TICULAR
970 INPUT "QUE SUJETO SE VA A COMPROBAR"; H$
980 T = 0
990 T = T + 1
1000 IF Z$(1, T) = H$ THEN 1040
1010 IF T < 25 THEN 990
1020 PRINT "NO TENGO DATOS ALMACENADOS EN "; H$
1030 GOTO 30
1040 K = 1
1050 K = K + 1
1060 IF Z$(K, T) <> "" THEN PRINT K; T; Z$(K, T)
1070 IF K < 25 THEN 1050
1080 GOTO 30
1090 REM **************
1100 REM INICIALIZACION
1110 CLS
1120 DEFINT A-Z
1130 DIM Z$(25, 25)
1140 RETURN
```



Sección segunda: LA INVESTIGACION

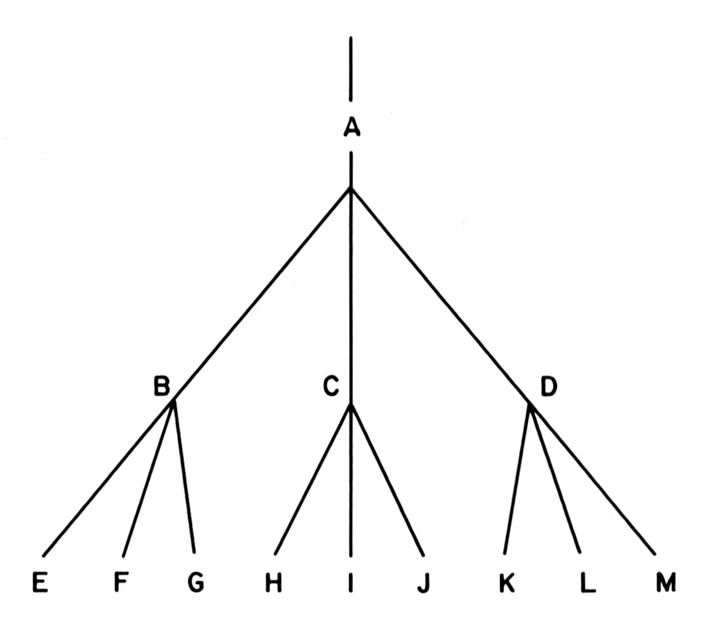
4
Arboles y RISITAS

En esta sección del libro, desarrollaremos un programa del tipo de LAS DAMAS llamado RISITAS. Lo usaremos para comentar algunas ideas sobre la investigación mediante árboles, en la cual el ordenador muestra cierta inteligencia investigando líneas de posibilidades relacionadas y escogiendo posteriormente la que juzga más adecuada.

Este tipo de investigación mediante árboles de posibilidades es común a la mayoría de los programas que implican solución de problemas. Algunas modificaciones, la mayoría de ellas muy importantes, tales como "podar" el árbol para ahorrarnos el tener que seguir ramas totalmente inútiles, o el llegar innecesariamente lejos en el caso de otras, son siempre necesarias para impedir que el proceso lleve demasiado tiempo, pero la idea básica de la investigación mediante árboles sigue siendo fundamental para la solución de problemas.

¿Por qué se les llama árboles?

Un árbol de búsqueda crece como cualquier otro árbol, salvo que al revés. Tomemos A en el siguiente diagrama como punto inicial de la investigación. Las "ramas" (llamadas B, C y D) que salen de éste, representan decisiones válidas (o jugadas correctas, si el programa es un juego de fichas). Las ramas más pequeñas que a su vez salen de éstas (E, F, etc.), son las implicaciones que acarrea el seguir esa rama.



Si este árbol representara un mecanismo de búsqueda de jugada en una partida de ajedrez, por ejemplo, A podría representar el movimiento de uno de los caballos. El programa sigue entonces las implicaciones que tendría esa jugada. B supone, pongamos por caso, que moviendo ese caballo se ataca una de las fichas del contrincante. La respuesta E es que el contrincante sencillamente retrocede esa ficha; F puede ser que el contrincante defiende esa ficha con otra; y G sería que el contrincante se come al caballo que le está atacando. E, F y G se dividirían posteriormente en N, O..., etc., las cuales cubrirían las respuestas posibles a cada una de esas acciones.

Puede verse que la investigación pasaría rápidamente a abarcar cada vez más posibilidades, llegando éstas a alcanzar proporciones astronómicas, de no haber algún medio de guiarla. Sólo un programa muy sencillo, como el del TRES EN RAYA, podría examinar todas las ramas de todos los árboles antes de escoger la siguiente jugada.

En el caso de otros programas, se examina cada rama hasta un punto predeterminado (seguidamente comentaremos cómo se determina ese punto) en vez de seguirla hasta el final, y el resultado de ese examen queda almacenado.

"Procesado paralelo"

Otra posibilidad consiste en examinar una rama superficialmente retrocediendo después para hacer lo mismo con las demás, una por una; tras esto se examinan con mayor profundidad las ramas que parecen más prometedoras. No merecería la pena, por ejemplo, seguir examinando una rama que partiera del supuesto de que el adversario sacrificaría gustoso su reina para comer un peón. Todas las ramas que impliquen —en opinión del mecanismo de evaluación del programa— un debilitamiento de la posición del adver-

sario, podrían abandonarse en el mismo momento de descubrirlo, dedicando el tiempo de procesado y el esfuerzo a seguir otros caminos más prometedores.

Cuando uno está desarrollando sus propios programas de inteligencia artificial, le merece la pena empezar a idearlos en términos de árboles de búsqueda, ya que lo más probable es que de un modo u otro tenga que utilizarlos. El árbol puede llegar a alcanzar un tamaño más que considerable, sobre todo si no se está trabajando en un dominio muy restringido (como lo hacemos nosotros en un programa del tipo de BLOQUELANDIA), o no se tienen muy claros los criterios que ha de seguir el programa a la hora de escoger entre las diferentes posibilidades.

He desarrollado, para esta sección del libro, un programa parecido a LAS DAMAS, cuyo nombre es RISITAS, a fin de demostrar algunos aspectos de la investigación mediante árboles. Como es natural, para entender el comentario que sigue, es necesario, en primer lugar, saber cómo se juega. A su debido tiempo, tendremos la ocasión de observar diversas instantáneas de una partida completa, pero, por el momento, basta con que veamos las primeras jugadas.

Este es el aspecto del tablero al inicio de una partida:

```
ORDENADOR:  0        PERSONA: 0

     12345678
     --------
   8  O O O O 8
   7 O O O O  7
   6  . . . . 6
   5 . . . .  5
   4  . . . . 4
   3 . . . .  3
   2  P P P P 2
   1 P P P P  1
     --------
     12345678
```

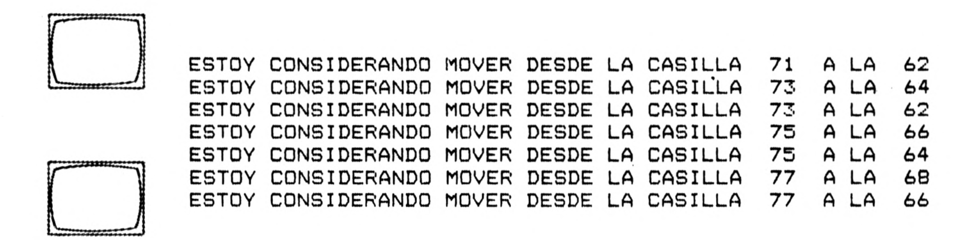
El marcador, a cero, aparece impreso por encima del tablero. Cada jugador empieza con ocho fichas (a diferencia de LAS DAMAS, que se inicia con 12). Las fichas del ordenador están en la parte superior del tablero (las O) y las del jugador humano en la parte inferior (las P, de ‘‘personas’’). El ordenador juega de arriba a abajo de la pantalla y el jugador humano, a la inversa.

Los puntos representan las casillas negras de un tablero de DAMAS. Las fichas se mueven igual que en las DAMAS, es decir, en diagonal de una casilla negra a otra. Cada ficha, pues, ocupa, de hecho, un punto que sólo aparecerá al mover esa ficha. Cada punto representa una posición a la que puede avanzarse una ficha.

Como dije previamente, cada ficha mueve como las fichas de las DAMAS, diagonalmente. El modo de comer también resulta familiar: se salta sobre una ficha del contrincante y se ocupa la casilla vacía inmediata a la que ocupaba la ficha comida. Sin embargo, al contrario de lo que sucede en las DAMAS, en este juego no hay saltos múltiples (no se pueden comer varias fichas a la vez).

El objetivo del juego es conseguir cinco puntos antes de que los consiga el contrincante. Hay dos maneras de marcar puntos. Una, como era de suponer, comiendo una ficha al contrincante. La otra, llegando al lado opuesto del tablero. En las DAMAS, la ficha que lo consigue, pasa a ser "dama" y puede avanzar o retroceder a su gusto. En este juego, la ficha desaparece al alcanzar la última hilera del lado opuesto (lo que significa, entre otras cosas, que en RISITAS ningún jugador tiene "damas" o, lo que es lo mismo, sus fichas no pueden retroceder en ningún caso).

Cuando se salta sobre una ficha del contrincante y la casilla situada inmediatamente después es una casilla de la última hilera del lado opuesto, se marcan dos puntos, en vez de uno. A su debido tiempo, en la partida de ejemplo, podremos ver cómo sucede esto. El ordenador nos dirá las jugadas que está considerando en cada momento de la partida, de modo que podremos ir siguiendo paso a paso cómo funciona la inteligencia de la máquina. Al principio del juego hay siete posibles aperturas, según se muestra en el tablero que hemos visto antes. El ordenador busca las jugadas correctas y cuando las encuentra las imprime en la pantalla antes de decidirse por una de ellas; de este modo:



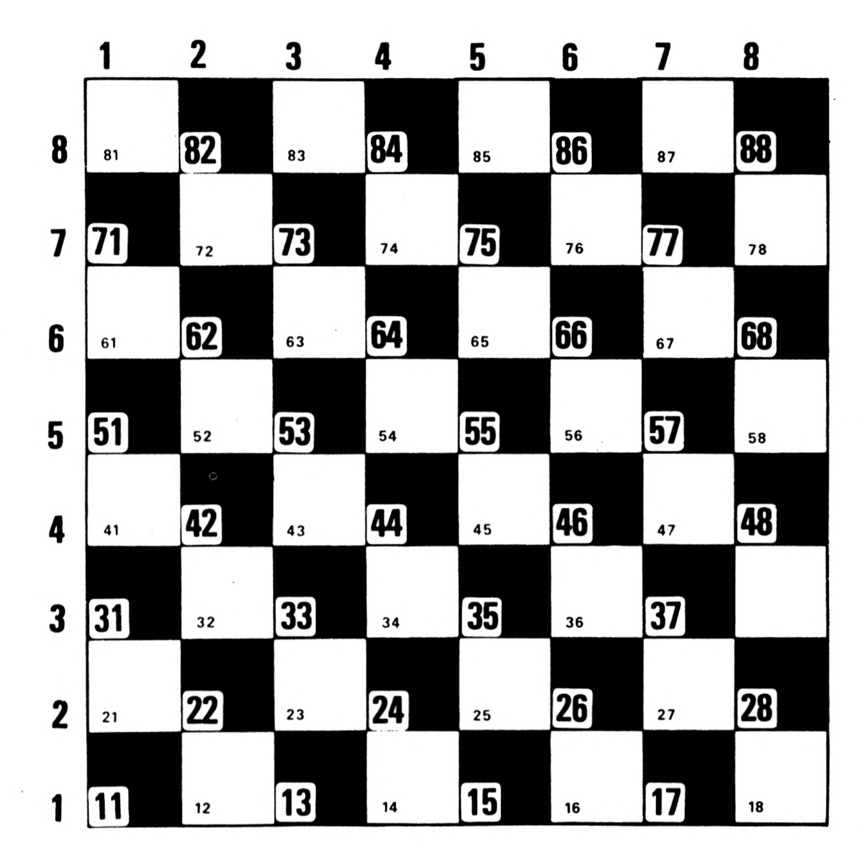
(el número de las casillas corresponde a la unión de uno de los números del margen izquierdo, en primer lugar, seguido de uno de los situados en el margen superior. Es decir, la casilla 71 es donde se unen la línea numerada 7, partiendo del margen izquierdo, con la numerada 1, partiendo de los números de la parte superior).

Los números que imprime el ordenador en este punto se refieren a aquellos incluidos en la matriz principal que contiene el tablero dentro del ordenador. El tablero numerado que utiliza el ordenador en RISITAS lo vemos en la página siguiente.

Se observará que los números no son consecutivos y ni siquiera empiezan por uno. No obstante, este tablero es más fácil de usar, en términos del ordenador, que aquellos en los que sólo las casillas negras están numeradas del uno al treinta y dos.

El ordenador necesita saber dónde están los límites del tablero y los números que faltan le proporcionan esa información. Por ejemplo, si intenta mover de la casilla 48 a la 59, el valor que tiene el elemento 59 en la matriz (cero, en el caso de RISITAS), le avisará de que haciendo esa jugada se sale del tablero.

La segunda, y mucho más importante, ventaja de este tablero, reside en la consistencia con la que se pueden especificar las jugadas, independientemente de en qué lugar del tablero sucedan. Explicaré un poco lo que quiero decir con esto. Para empezar, observemos la lista de jugadas que está considerando el ordenador y tomemos en cuenta la sen-



cilla relación matemática que conecta la casilla origen con la casilla destino (desde la que se mueve a la que se mueve):

```
71  A LA  62    - 9
73  A LA  64    - 9
73  A LA  62    - 11
75  A LA  66    - 9
75  A LA  64    - 11
77  A LA  68    - 9
77  A LA  66    - 11
```

La diferencia entre ambas es o menos nueve o menos once. Y si se comparan estos números con el tablero, veremos quė las jugadas hacia abajo y hacia la izquierda son siempre menos once y aquéllas hacia abajo y hacia la derecha son siempre menos nueve.

Esto es cierto para todo el tablero. Cualquier jugada del ordenador que no implique comer una ficha al adversario, ha de ser, partiendo de la casilla inicial, o menos nueve o menos once. Estoy seguro de que el lector sabrá apreciar cuán cómodo es esto desde el punto de vista del ordenador. (Los incrédulos pueden intentarlo utilizando un tablero que tenga solamente numeradas las casillas negras del uno al treinta y dos; no tardarán en darse cuenta de los inmensos problemas que se crean.)

Es más, con este tablero, al ordenador le resulta bastante fácil tomar decisiones. Pongamos que la casilla en la que se encuentra el ordenador tiene el número X. Si hay una ficha del jugador humano en la casilla X-9 y la X-18 está vacía, éste sabe que puede comer saltando a la casilla X-18. Se marca un punto en el marcador y X-9 pasa a ser una casilla vacía.

Y hay todavía algo más: el ordenador tiene en cuenta lo que puede suceder después de esa jugada; piensa primero en las jugadas que puede hacer el jugador humano a continuación y, entre éstas, cuál es la más probable. Es en este punto donde realmente hace su aparición la "inteligencia" del ordenador. Si hay una ficha del jugador humano en la casilla X-27, el ordenador puede suponer —posiblemente sin que le falte razón— que la siguiente jugada del juegador humano será comer la ficha del ordenador situada en la casilla X-18, moviendo para ello a la casilla X-9.

La posición después de haber comido (recordemos que estamos, considerando ahora que somos el ordenador, en la casilla X-18) se encuentra también amenazada potencialmente por la casilla X-25, de estar libre la X-7. Llegados a este punto la explicación puede parecer un poco sorprendente, conque lo mejor es que intentemos seguirla en el tablero que aparece unas páginas más atrás o en un tablero de damas numerado del mismo modo.

El ordenador se da cuenta, asimismo, de cuándo se encuentra atacada una de sus fichas. Volvamos a imaginarnos que somos el ordenador y que tenemos una ficha en la casilla X. El jugador humano mueve a la casilla X-9. Sabemos que X+9 está libre, de modo que el jugador humano podrá mover a ésta comiéndonos a nosotros que estamos en la X. Para impedirlo podemos, ya sea poner una de nuestras fichas en X+9 o, si esto no fuera posible, mover otra ficha de modo que la casilla X+9 quede amenazada. Con esto quizá logremos convencer al jugador humano de que es mejor que no coma esa ficha.

En este juego, al contrario de lo que sucede en las DAMAS, uno no está obligado a comer llegado el caso y puede dejar de hacerlo si no lo considera adecuado. Uno puede preferir no comer en un momento determinado una ficha amenazada porque sabe que si espera un poco puede marcar dos puntos comiendo esa misma ficha y llegando al mismo tiempo a la última hilera del tablero.

En profundidad

El árbol de búsqueda de RISITAS no llega a profundizar mucho, aunque el programa se las arregla para jugar razonablemente bien y ganar no pocas partidas (incluyendo la que utilizo un poco más tarde en este mismo capítulo para explicarlo). Habrá quien se sienta tentado a pensar que el programa podría jugar perfectamente si se investigaran completamente todas las posibilidades que ofrece cada árbol.

RISITAS es un juego menos complejo que las DAMAS; no hay en él saltos múltiples ni se pueden hacer "damas" y, por tanto, no estaríamos muy descaminados al pensar que no es del todo imposible desarrollar un sistema perfecto. Por lo menos, debería ser posible crear un camino mediante el cual el ordenador, con sólo seguirlo, pudiera jugar extremadamente bien.

Podemos hacerlo siguiendo un método en cierto modo similar al del TRES EN RAYA jugado por el "ordenador-caja de cerillas" del que hablábamos en el capítulo de-

dicado al programa de TICTAC. Es decir, podemos examinar todas las bazas posibles, de todas las partidas posibles, analizándolas en profundidad. Después de todo, disponemos de unos ordenadores infatigables y nada nos impide que sean ellos quienes lleven a cabo los "trabajos pesados".

Pensemos un poco sobre esto. Sabemos (porque el ordenador nos lo ha dicho antes) que son siete las jugadas que puede hacer al principio de la partida. Así pues, nuestro árbol, con A arriba a modo de tronco, empieza con las ramas B, C, D, E, F, G y H. El jugador humano puede escoger igualmente entre siete jugadas diferentes al inicio de la partida. A cada una de las siete ramas iniciales habrá que añadirle, pues, siete subramas (o "nudo", que es como se llama al punto de intersección de las ramas). Después de que cada jugador haya movido una vez, e incluso antes de que el programa empiece a considerar las posibles respuestas que puede dar a la primera baza del jugador humano, ya son cuarenta y nueve las diferentes ramas que hemos de seguir.

La situación empeora a continuación. Ahora que ambos jugadores han movido una ficha de su primera hilera, son dos las jugadas realizables (en algunos casos una, si la jugada inicial se ha hecho con una de las fichas de los extremos) más otras seis (posiblemente la primera jugada ha bloqueado otra realizable con una ficha que está todavía en la primera hilera). Esto significa que ahora son ocho multiplicadas por cuarenta y nueve las ramas que hemos de considerar, y esto antes de que el jugador humano haya movido por segunda vez.

Un árbol de búsqueda de este tipo aplicado a las DAMAS contendría tantos nudos como diez elevado a cuarenta. Considerando una frecuencia de tres millones de nudos por segundo (lo que necesitaría un ordenador estupendísimo), el examen total de este árbol llevaría un número de años que sería igual a diez elevado a veintiuno.

Sugeríamos un poco antes que una manera de podar el árbol era abandonar sencillamente las ramas poco aprovechables (como por ejemplo, todas aquellas posibilidades en las que el contrincante ponga deliberada e innecesariamente en peligro una de sus fichas), con lo que ahorrábamos tiempo y esfuerzo para examinar otras ramas que merecieran más la pena. Asimismo, sugeríamos que el ordenador podría ir comprobando una por una todas las ramas, pero sin profundizar mucho en ninguna de ellas, con la posibilidad de abandonar las más débiles para concentrarse en las más prometedoras.

Para hacer esto, hemos de poder asignar un valor a la posición encontrada. Este valor puede ser un número (basado en algo parecido al del programa de DAMAS de Samuel del que hablamos en el capítulo dedicado a TICTAC) o puede basarse en un esquema jerárquico en el que se ordenen las jugadas escogidas decidiendo no seguir la mayoría de las ramas que podrían generarse. Como veremos enseguida, esto es lo que hicimos en el programa de RISITAS.

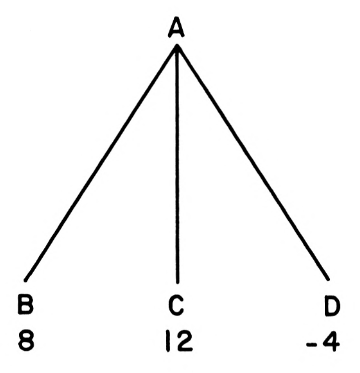
Mini-maxi

No obstante, primero hemos de profundizar un poco más en el estudio de los árboles de búsqueda, si lo que queremos es conseguir que el ordenador juegue perfectamente. RISITAS utiliza una modalidad de la técnica conocida con el nombre de "mini-maxi", con la cual se pueden ir podando las ramas que aparecen sin cesar.

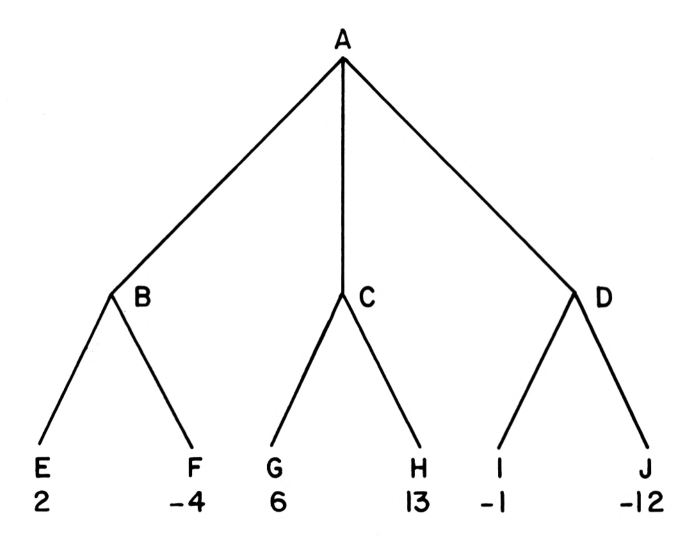
Para usar esta técnica, sin embargo, el ordenador tiene que estar capacitado para asignar valores numéricos a las posiciones que va descubriendo.

Imaginemos que son tres las posibilidades que está considerando y que cada una de ellas consiste en una jugada con una ficha diferente. El valor dado a esa jugada podría depender, en parte, de la proximidad de la ficha con respecto al centro del tablero después de haberla movido, de si amenaza (inmediatamente o tras otra jugada) una ficha del adversario, de si la casilla que se está tomando en consideración se encuentra atacada por una ficha enemiga, de si con esa jugada se come una ficha enemiga o, finalmente, de si se consigue cualquier otro objetivo (como llegar a la última fila del lado opuesto).

Este sería el árbol, con las jugadas B, C y D en los extremos de las tres primeras ramas y su puntuación (el valor que tienen) bajo ellas:



Podemos ver que C tiene el valor más alto, de modo que la elección parece obvia. Recordemos que este arbolito está basado en la situación una vez que el ordenador ha movido. Sin embargo, si la máquina examinara la siguiente serie de ramas, en la que se consideran y evalúan las posibles respuestas del jugador humano, vería esto:



Los valores que se dan aquí a los nudos E a J están calculados en términos de la evaluación de las posiciones del tablero por parte del jugador. La mejor jugada que podría

hacer el ordenador sería aquella que hace que las posibilidades del jugador humano, tras haber movido, en respuesta a la jugada del ordenador, sean las que le dejan en una posición más débil para afrontar las siguientes jugadas. El ordenador elegirá, pues, aquella posibilidad con la que marque más puntos al tiempo que deje minimizadas las posibilidades del jugador humano. De aquí viene el término "mini-maxi".

Si consideramos que el ordenador no va a profundizar más en su investigación con el fin de evaluar su propia posición después de cada jugada del jugador humano (y posiblemente para evaluar las respuestas a esa respuesta), probablemente escogerá la jugada B. Esta le deja en una posición bastante fuerte (con un índice de 8), aunque no tanto como en el caso de la jugada C (con un índice de 12).

El ordenador parte de la base de que el jugador va a hacer la mejor jugada que pueda dentro de lo que le permitan las circunstancias. De haber escogido el ordenador la jugada C, con el fin de conseguir el índice más alto inmediatamente después de la jugada, habría dejado al jugador humano la posibilidad de escoger la jugada H, con lo que éste terminaría con un índice de 13. En vez de esto, al escoger el ordenador la jugada B, todo lo que puede hacer el jugador humano es responder escogiendo el nudo E, con lo que se queda con un índice de 2.

Decíamos antes que RISITAS funciona asignando un valor a todas las jugadas posibles, siguiendo un orden jerárquico. Escoge las jugadas en relación con esa jerarquía. De serle posible, siempre hará la jugada situada más arriba.

En un grado u otro siempre está presente la técnica "mini-maxi". El programa piensa solamente en términos de ventaja material, es decir, intenta reducir el número de fichas del contrincante al tiempo que preserva las propias.

Por ejemplo, el programa ve dos posibles capturas; con una de ellas se expone a que después le coman a él una ficha. Naturalmente hará la jugada que le deja con la posición más fuerte después de comer (conservando en el tablero la ficha con la que ha comido) e ignorará la jugada que afianza la posición de su contrincante (ya que éste, a su vez, comería una ficha del ordenador).

La jerarquía de jugadas que utiliza RISITAS para podar el árbol de "posibles jugadas" y ahorrarse el tener que investigar ramas que representan jugadas que es muy poco probable que lleve a cabo, es la siguiente. Se almacenan todas las jugadas encontradas que se ajusten a la descripción:

- Capturas seguras que además amenazan otras fichas del jugador humano y no exponen a ninguna ficha propia.
- Capturas que dejen las fichas que han efectuado esa captura en situación de seguridad frente a las fichas del contrincante.
- Otras capturas.
- Jugadas que protejan las fichas amenazadas.
- Rechazo al azar de las jugadas citadas si por el hecho de efectuar la jugada se exponen fichas propias.
- Jugadas con las que se llegue a la última hilera del lado opuesto aunque no se coma ninguna ficha del contrincante.
- Jugadas que no expongan ninguna ficha del ordenador aunque éste no coma con ellas ninguna ficha del contrincante.
- Cualquier jugada correcta.

En el momento en que encuentra una jugada con la que puede comer alguna ficha del contrincante sin exponer por ello las suyas, el ordenador no se preocupará en seguir investigando el árbol de jugadas. En efecto, poda automáticamente las ramas con nudos "bajos" mediante el sencillo método de ni siquiera tenerlas en cuenta. Esto puede parecer un tanto imprudente y precipitado y, de hecho, significa que el programa no es capaz de jugar siguiendo ningún tipo de estrategia global, pero en la práctica funciona sorprendentemente bien (a lo que contribuye, desde luego, la sencillez del juego) y se las arregla para dar la impresión de que tiene una técnica de lo más refinado.

Espero que el lector pueda apreciar que esta ordenación jerárquica de la jugada disminuye el número de posibilidades que deberían investigarse. Al examinar más detenidamente el programa veremos que éste, en primer lugar, peina el tablero, casilla por casilla, buscando posibles capturas; capturas que posteriormente almacena como "buena captura, segura", "segura" o "captura".

Si las zonas de almacenamiento siguen vacías cuando ha terminado de peinar el tablero, el ordenador vuelve a hacerlo buscando ahora si alguna de sus fichas se encuentra amenazada por alguna ficha del jugador humano.

Si en esta búsqueda tampoco consigue encontrar una jugada que le convenga, nuestro ordenador mira si tiene alguna ficha en la penúltima hilera del lado opuesto del tablero con el fin de, en el caso de tenerla, moverla a la última hilera marcándose así un punto. Para llevar a cabo esto cuenta con un orden predeterminado que le garantiza que si son dos las fichas con las que puede mover para situarse en la última hilera del tablero, moverá primero la que esté situada más cerca del medio del tablero, en la suposición de que siempre es más probable que ésta llegue a encontrarse atacada por el contrincante que una ficha situada en un extremo del tablero. Se trata de una suposición un poco sobre la marcha, pero garantiza que el ordenador no se limitará a mover la primera ficha que encuentre en la penúltima hilera sin antes ver cuál tiene más posibilidades de ser comida por una ficha enemiga.

Si todavía no ha conseguido mover, el ordenador vuelve a peinar el tablero almacenando todas las jugadas seguras que descubra (es decir, cualquier jugada que no exponga a una posterior captura por parte del contrincante a la ficha con la que se ha movido). Posteriormente el ordenador escoge una al azar entre todas ellas (en el caso de haber encontrado más de una).

Si en esta búsqueda tampoco consigue encontrar una jugada, el ordenador busca cualquier jugada, siempre que sea correcta, para lo que va escogiendo al azar las diferentes posiciones de las fichas en el tablero. Si pese a los 200 intentos que se permite a sí mismo, no consigue encontrar una jugada correcta, el ordenador se dará por vencido. En breve pasaremos revista a las partes más importantes del listado identificando las subrutinas que llevan a cabo cada una de las tareas que aquí hemos especificado.

Posiblemente habrá quien piense que los múltiples barridos del tablero son, en cierto modo, una pérdida de tiempo. ¿Acaso no puede el programa realizar los diferentes niveles de búsqueda con un solo barrido? La respuesta es, por supuesto, "sí", salvo que en muchos casos esto significaría un despilfarro considerable ya que buscaría y almacenaría jugadas que ni siquiera tiene la intención de examinar. De todos modos, puede que el lector quiera modificar el programa, o escribir uno nuevo, con el fin de que todas las comprobaciones se hagan en un solo barrido; de hacerlo, podrá ver los efectos que esto tiene en el tiempo de reacción del programa nuevo o modificado.

Es obvio decir que el sistema jerárquico que determina el valor relativo de las jugadas

podría combinarse, para conseguir una mayor flexibilidad, con una función de evaluación. Esto introduciría, al igual que hizo Samuel con su programa, cosas tales como el número de fichas con que cuenta un jugador en comparación con las que cuenta el otro, el número de piezas que se encuentran directamente amenazadas y el "control del centro", defínase como se defina.

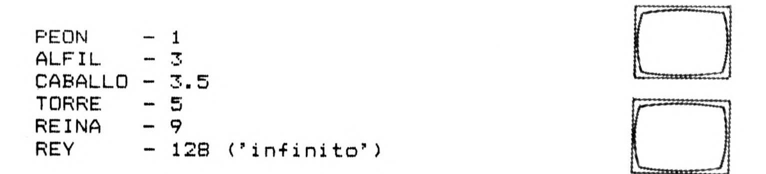
Una alternativa al uso de la función de evaluación sería que el ordenador almacenara todas las posiciones posibles del tablero asignándoles un valor predeterminado a cada una de ellas. Pero esta solución, al igual que la idea de hacer árboles completos que cubrieran todas las consecuencias y resultados posibles de todas las jugadas de cada partida, choca contra la barrera de los números astronómicamente altos. En las DAMAS hay unas diez elevado a cuarenta posibles posiciones del tablero y las de RISITAS no le van muy a la zaga.

No existen reglas que puedan aplicarse a la hora de desarrollar las funciones de evaluación para los programas de juegos de tablero. La función inicial vendrá guiada por un trabajo de intuición "informado", tras lo cual se probará esa función en la práctica, lo que permite modificarla hasta que actúe correctamente. La ventaja de los juegos sencillos, como el TRES EN RAYA, es que el programa puede jugar una y otra vez contra un contrincante que juegue al azar, que no sepa jugar. Los resultados de estas partidas pueden utilizarse entonces para modificar automáticamente la función de evaluación; o también, pueden jugarse un gran número de partidas con una versión determinada de la función comparándolas posteriormente con un número similar de partidas jugadas con otras versiones de la misma. No es nada sencillo programar un contrincante que juegue una y otra vez contra el programa cuando se trata de juegos más complejos, como el AJEDREZ, las DAMAS o, incluso, RISITAS.

Elementos de peso

La tarea se simplifica por el hecho de que algunos de los elementos que integran una función de evaluación, generalmente están equilibrados al estar multiplicados por un valor más alto que otros. La modificación de la función de evaluación puede ser entonces sólo una cuestión de modificar los factores de peso, más que de tener que añadir o suprimir completamente nuevos elementos.

Intentaré explicar el párrafo precedente con un ejemplo concreto. Los jugadores de ajedrez saben por experiencia que el valor relativo de las piezas puede expresarse, a grandes rasgos, del modo siguiente:



Podemos crear la primera evaluación sencillamente sumando las fichas que tenemos y restando las que tiene nuestro contrincante, con lo que damos la medida de nuestra "fuerza" relativa; de este modo:

```
Fuerza = n*wP + 3*n*wB + 3.5*n*wK +
5*n*wR + 9*n*wQ - (N*bP + 3*N*bB
+ 3.5*N*bK + 5*N*bR + 9*N*bQ)
```

Con ésta como evaluación inicial, podríamos escribir un programa de AJEDREZ sin grandes refinamientos técnicos; un programa que no tendría inconveniente en sacrificar sus fichas, o cambiarlas, cuando su "fuerza" fuera positiva y que sería más conservador a este respecto cuando la "fuerza" fuera negativa. Jugando contra este programa y usando esta función para ayudar a decidir qué ramas deben investigarse (utilizando la técnica "mini-maxi") podríamos darnos cuenta de que, en realidad, a la torre le hemos dado un valor demasiado bajo, lo que nos lleva a cometer errores innecesarios. Así pues, podríamos aumentar su valor a 6, 5 ó 7.

El valor de la función de evaluación podría incrementarse si se pudiera incorporar la movilidad (expresada como el número de movimientos que tiene cada ficha). El valor de la torre, por ejemplo, podría expresarse como 5*n*wr+3*rm (siendo rm igual a los movimientos que podría hacer). Y todavía podríamos llevar a cabo una elaboración posterior de esta función sencillamente añadiéndole al valor de la ficha un número que reflejara el "valor" de la casilla que ocupa (dándole, pongamos por caso, un valor de 8 a las cuatro casillas centrales, 6,5 a las casillas que rodean a las cuatro centrales y 4 a las siguientes). Y así sucesivamente, podría irse haciendo una función cada vez más elaborada. Cuando uno se detiene a pensar en los problemas inherentes a la creación de una función de evaluación para un programa tan complejo como puede serlo el AJEDREZ, se da perfecta cuenta de que una tarea de este tipo es algo mucho más complicado de lo que hubiera podido imaginar.

Quien se sienta interesado en el desarrollo de funciones de evaluación, podría empezar desarrollando una para RISITAS y usándola para modificar el modo en que se escogen en éste las jugadas. Encontrará que incluso una función no muy bien acabada —si consigue que el ordenador la aplique en la práctica— mejorará notablemente el juego de éste.

Sería posible, de ser casi infinitos el tiempo y el poder informático con que contamos, investigar todas las ramas del árbol hasta llegar al final de la partida. Esto significaría que tendríamos que investigar una cantidad enorme de posibilidades, como veremos enseguida. Quizá, una aproximación más sensata sería limitar la profundidad de la investigación. Por el momento, admitamos que hemos decidido deliberadamente dar tan sólo dos pasos en nuestra investigación del árbol; una jugada y la posible respuesta a esa jugada por parte del contrincante.

Una investigación de este tipo recibe el nombre de "2-capas" porque estamos investigando en profundidad sólo una jugada y la inmediata respuesta a ésta. A grandes rasgos, RISITAS utiliza un tipo de investigación de "2-capas" (pero sin una técnica de "mini-maxi" global) ya que intenta encontrar la jugada que le proporcione la mejor ventaja material suponiendo que el contrincante, en respuesta, hará también la mejor jugada posible en términos materiales (es decir, si puede, hará una captura). Si admitimos que la función de evaluación que hemos elaborado es realista, cuanto más profunda sea la capa, mejores serán los resultados que consiga el programa.

No obstante, a medida que vayamos profundizando en la investigación, volveremos a encontrarnos con cantidades astronómicas. Si admitimos que en el TRES EN RAYA

(o CEROS Y CRUCES) hay tres jugadas posibles al inicio del juego (es decir, una jugada realizada en una esquina es igual a cualquier otra jugada realizada en otra esquina, ya que el primer tablero puede transformarse en los otros mediante una simple rotación), en el nivel 2-capas habrá entonces doce posiciones y una cantidad que se aproxima a 12×7 en el siguiente nivel (y decimos "que se aproxima" porque no todas las partidas completarán estas posibilidades, ya que se podría llegar a un empate o a una victoria antes de que se cumplieran las nueve posiciones).

En otros juegos las posibilidades aumentan todavía más dramáticamente. En el AJEDREZ, por ejemplo, una investigación con un promedio de 4-capas tiene que vérselas con una cantidad de posibilidades que ronda el millón.

El algoritmo alfa-beta

¿Cómo podemos arreglárnoslas con todas esas cifras cuando estamos intentando escribir un programa que juegue razonablemente bien pero que no tarde un número de años igual a diez elevado a cuarenta en hacer una jugada? Es el momento de introducir el algoritmo alfa-beta, una ayuda de lo más eficaz a la hora de desbastar las ramas de nuestro árbol.

El algoritmo alfa-beta es muy sencillo, pero muy poderoso. Dice que —en el caso de poder escoger entre un conjunto de jugadas posibles— una vez que hemos encontrado la jugada que se ajusta a nuestras necesidades (y nuestras necesidades pueden expresarse en términos de mejorar la puntuación dada por la función de la evaluación), no hay necesidad de seguir buscando otra jugada en ese juego.

El algoritmo alfa-beta recibe este nombre porque opera sencillamente siguiendo de cerca dos valores llamados alfa y beta. Nuestro programa investiga un árbol de posibilidades intentando encontrar una buena jugada. Alfa es el valor de la mejor jugada que éste ha descubierto hasta el momento. Al seguir investigando, el programa encuentra una jugada que produce un valor más bajo que alfa. Inmediatamente sabe que no merece la pena seguir esa rama porque le llevaría a un resultado peor que el que ya ha encontrado. Esto significa que el ordenador queda en libertad para seguir buscando en una rama nueva.

Mientras tanto, el programa calcula asimismo las posibles respuestas a sus jugadas. Si encuentra una respuesta mala desde el punto de vista del contrincante —de modo que no es muy posible que éste haga esa jugada—, decide que no tiene sentido seguir las situaciones que podrían derivarse de esa respuesta. Beta es el valor que tiene el contrincante cuando hace la mejor jugada posible en respuesta a otra del ordenador. La investigación se interrumpe si la rama lleva a una jugada del adversario que disminuya el valor de beta, considerada desde el punto de vista del jugador humano.

La interrupción en la investigación debida al descubrimiento de que la rama que está investigando baja la puntuación del ordenador se llama "interrupción alfa". La otra interrupción, la que se produce cuando la rama lleva a una jugada del contrincante que disminye el valor de beta, se llama, como era de esperar, "interrupción beta".

En la siguiente secuencia podemos ver cómo actuaría una forma, no especialmente cuidada en este caso, de la cara alfa de este algoritmo:

— cálculo del valor del tablero;
— búsqueda de la primera jugada;
— cálculo del valor del tablero tras la primera jugada;
— búsqueda de la mejor respuesta por parte del contrincante y cálculo del valor que tendrá el tablero después de esa jugada;
— almacenamiento de ambos valores;
— búsqueda de la siguiente jugada realizando el mismo proceso;
— si la nueva jugada produce un resultado "mini-maxi" mejor que la anterior, el ordenador descarta la primera y almacena sólo la segunda;
— continuación del proceso manteniendo sólo almacenada la jugada que mejor resultado "mini-maxi" ha producido hasta el momento.

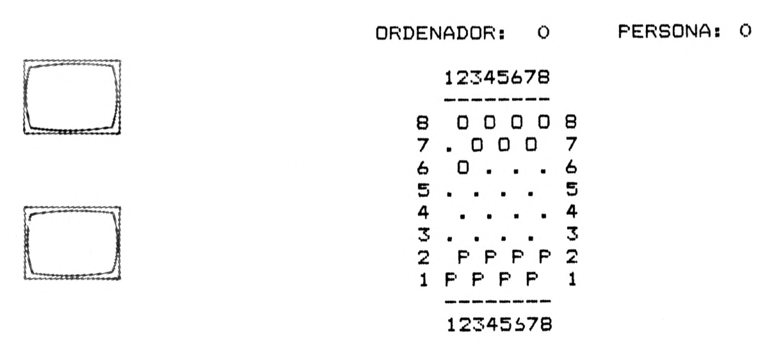
Siguiendo este proceso el ordenador terminaría con una sola jugada que —teniendo en cuenta que la búsqueda es limitada— sería la "mejor" jugada de todas las que pudiera hacer.

Se ha de observar que el algoritmo alfa-beta puede aplicarse en muchas áreas de toma de decisiones y no sólo en los juegos de tablero. Muchos programas inteligentes, de los que implican una elección entre un número determinado de posibilidades, siguen una línea alfa-beta para determinar cuál es la mejor elección de cara a la acción.

Volvamos ahora a nuestro programa RISITAS. Recordemos que al principio del capítulo observamos la posición de apertura del tablero y que el ordenador generó una lista de las jugadas que estaba considerando. Las jugadas eran las siguientes:



El ordenador determina que todas éstas son jugadas en las que no se hace ninguna captura pero son "seguras" e iguales en cuanto a su valor, por tanto, escoge una de ellas al azar y mueve, como vemos, de la casilla 71 a la 62:



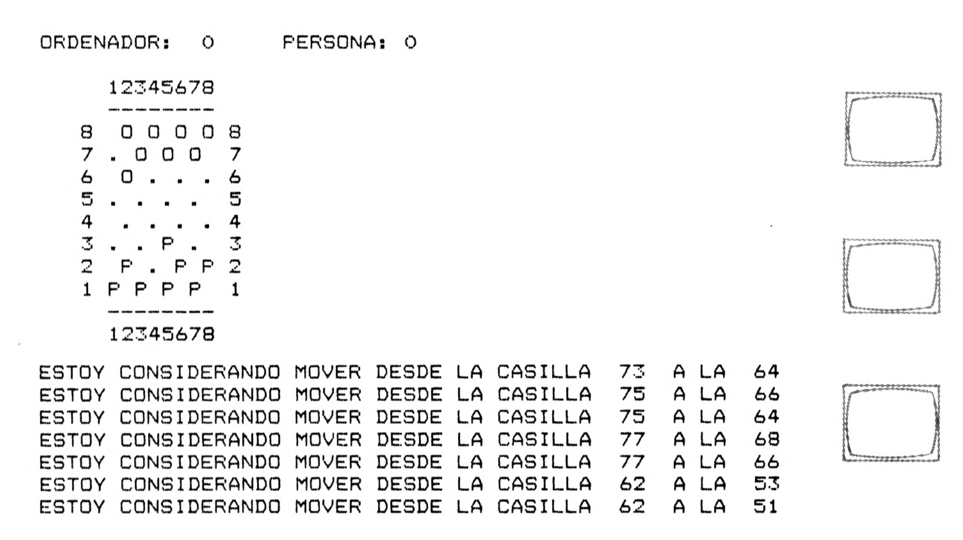
De hecho, el valor de las jugadas puede no ser el mismo en todos los casos, ya que el hecho de avanzar hacia el centro del campo, siempre que sea posible hacerlo, tiene un valor en sí mismo, pero el programa carece de la información necesaria, porque no se la

hemos facilitado, para hacer una valoración a este respecto y, por tanto, cree que todas las jugadas son iguales (y actúa de acuerdo con esa creencia).

Se introduce entonces la respuesta del jugador humano, utilizando un número, compuesto por uno de los de las hileras verticales seguido de otro de los marcados en la parte inferior del tablero (o en la superior), para marcar la casilla "desde la" que se va a mover y luego, cuando éste haya aparecido, siguiendo el mismo procedimiento para marcar la casilla "a la" que se va a llegar:

```
MUEVO DESDE LA CASILLA?  24
          A LA CASILLA?  35
```

El tablero vuelve a imprimirse y el ordenador revela las jugadas que está considerando:



```
ORDENADOR:  0      PERSONA: 0

     12345678
     --------
   8  O O O O 8
   7 . O O O  7
   6  O . . . 6
   5 . . . .  5
   4  . . . . 4
   3 . . P .  3
   2  P . P P 2
   1 P P P P  1
     --------
     12345678

ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  73  A LA  64
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  75  A LA  66
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  75  A LA  64
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  77  A LA  68
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  77  A LA  66
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  62  A LA  53
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  62  A LA  51
```

Unas cuantas jugadas después, el tablero presenta esta situación (en esta impresión no se muestran todas las posibles jugadas del ordenador):

```
ORDENADOR:  0      PERSONA: 0

     12345678
     --------
   8  O O . O 8
   7 . . O .  7
   6  . O O O 6
   5 O . . .  5
   4  . . . . 4
   3 P P P P  3
   2  P . . P 2
   1 . . P P  1
     --------
     12345678

ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  82  A LA  73
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  82  A LA  71
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  64  A LA  55
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  64  A LA  53
```

El ordenador no puede comer ninguna ficha al adversario y todas las jugadas posibles pueden clasificarse como "buena, segura". El ordenador sigue jugando, escogiendo al azar entre las jugadas que ha descubierto y almacenado como igualmente buenas (al menos, bajo su propio punto de vista):

```
ORDENADOR:  O      PERSONA: O

  12345678
  --------
8  O O . O 8
7 . . O .  7
6  . . O O 6
5 O . O .  5
4  . . . . 4
3 P P P P  3
2  P . . P 2
1 . . P P  1
  --------
  12345678

MUEVO DESDE LA CASILLA?  37
           A LA CASILLA?  48
```

```
ORDENADOR:  O      PERSONA: O

  12345678
  --------
8  . O . O 8
7 . O O .  7
6  . . O O 6
5 O . O .  5
4  . . . P 4
3 P P P .  3
2  P P . P 2
1 . . . P  1
  --------
  12345678
```

La situación se hace ahora un poco más compleja. Las fichas del adversario están muy cerca unas de otras, de modo que existe la posibilidad de mover a una situación de peligro. Por ejemplo, si el ordenador moviera de la casilla 66 a la 57, lo más probable es que la respuesta del jugador humano fuera comer esa ficha utilizando su ficha de la casilla 48. El mecanismo de búsqueda del programa descubre y revela esto. Recordemos que el programa intenta continuamente asegurarse de que su puntuación después de cada jugada sea la mejor posible y, al mismo tiempo, de que no deja al jugador humano la posibilidad de aumentar su puntuación. El mover de la casilla 66 a la 57 iría en contra de estos dos objetivos, de modo que un programa que procediera en este sentido e hiciera esta jugada —dadas todas las jugadas alternativas que, de hecho, existen en este momento— sería un programa de lo más pobre.

Afortunadamente, RISITAS sí que es capaz de darse cuenta de unos peligros tan ele-

mentales y, por tanto, mueve de la 73 a la 62, que es una jugada clasificada como "buena, segura".

```
ORDENADOR:  0      PERSONA: 0

  12345678
  --------
8  . O . O 8
7 . . O .   7
6  O . O O 6
5 O . O .   5
4  . . . P 4
3 P P P .   3
2  P P . P 2
1 . . . P   1
  --------
  12345678

MUEVO DESDE LA CASILLA?  17
          A LA CASILLA?  26
```

```
ORDENADOR:  0      PERSONA: 0

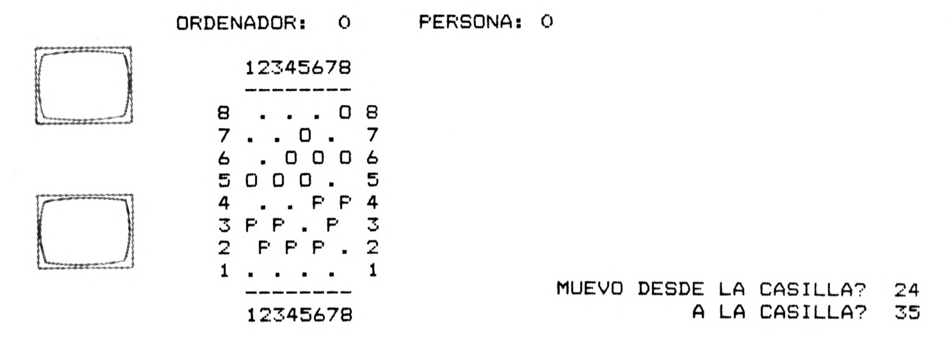
  12345678
  --------
8  . O . O 8
7 . . O .   7
6  . . O O 6
5 O O O .   5
4  . . . P 4
3 P P P .   3
2  P P P P 2
1 . . . .   1
  --------
  12345678

MUEVO DESDE LA CASILLA?  28
          A LA CASILLA?  37
```

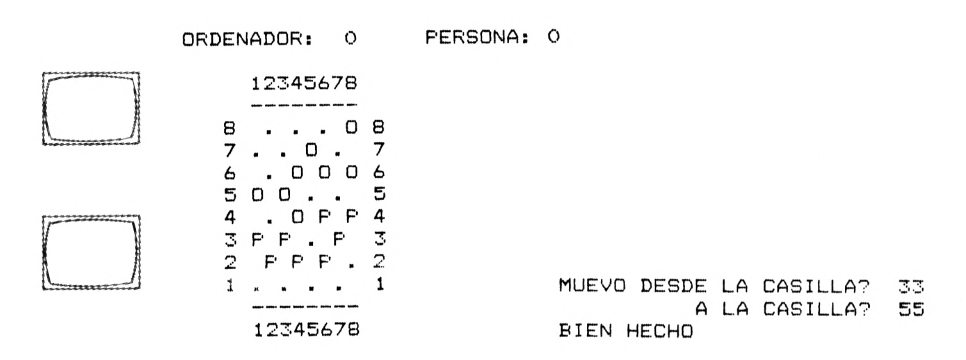
Y la partida continúa:

```
ORDENADOR:  0      PERSONA: 0

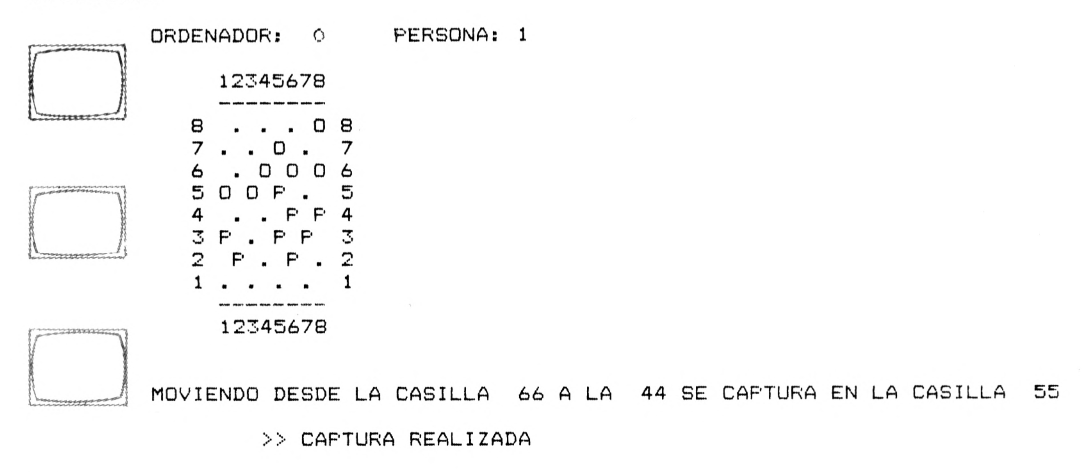
  12345678
  --------
8  . O . O 8
7 . . . .   7
6  . O O O 6
5 O O O .   5
4  . . P P 4
3 P P . P   3
2  P P P . 2
1 . . . .   1
  --------
  12345678
```



Son muy pocas ahora las jugadas fáciles ("buena, segura") de que dispone el programa y hace una, avanzando desde la casilla 55 al peligro que supone la 44 (por parte de la ficha del contrincante situada en la casilla 33), con la que parece cometer su primer error. El jugador humano responde comiéndole esa ficha, por lo que recibe la más sincera felicitación del ordenador.

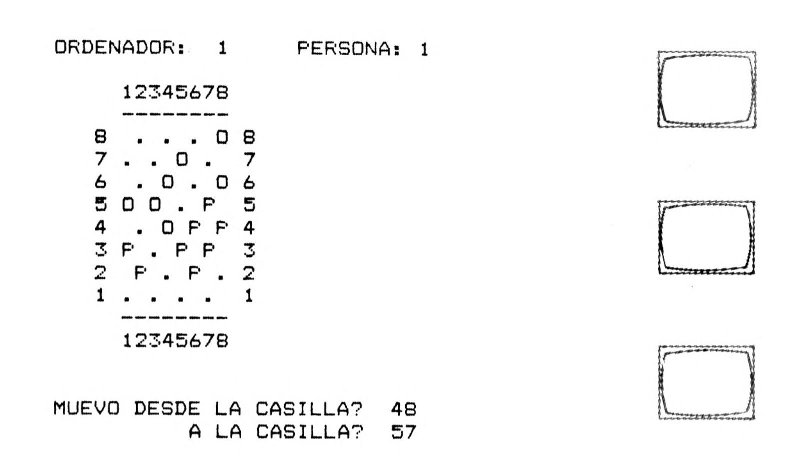


Por supuesto, el jugador humano ha movido a una posición peligrosa, un hecho que RISITAS observa enseguida. Informa de la jugada que ha encontrado y que está a punto de realizar.



El marcador está ahora empatado a uno. Recuerde que se marca un punto por cada ficha del contrincante que se captura o por cada ficha propia que uno consiga colocar en la última hilera del lado opuesto del tablero. En breve veremos en acción este segundo método de marcar puntos.

Afortunadamente, el programa, después de haber comido, no se ha situado en una posición en la que pueda ser comido a su vez:



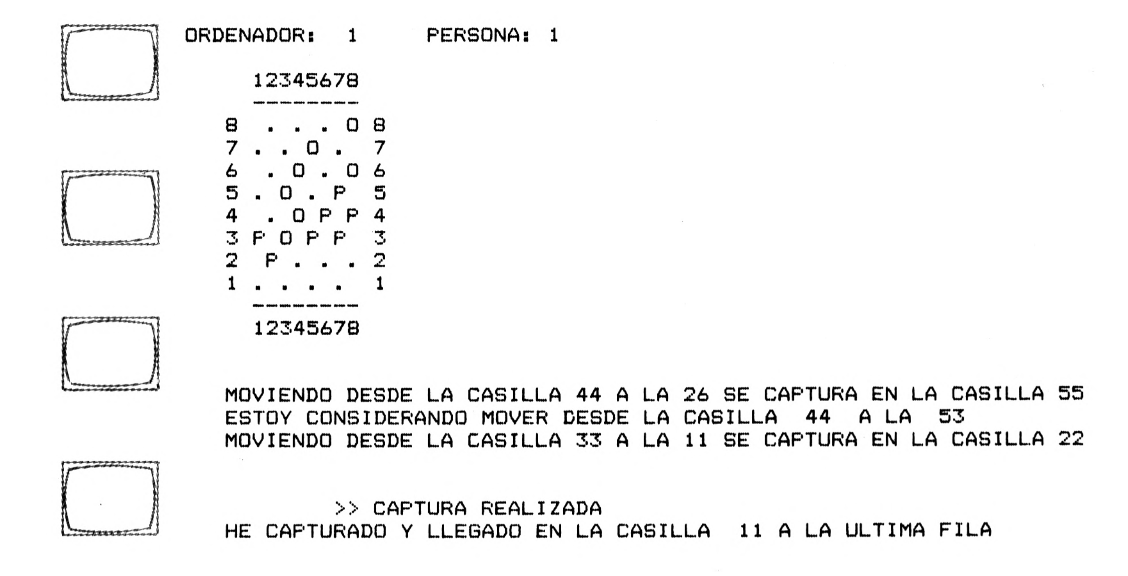
El programa sabe que puede avanzar hasta ponerse al lado de una ficha enemiga, sin correr ningún riesgo por ello, puesto que cuenta con toda una retaguardia que impide que aquélla pueda comerle; así pues, mueve de la casilla 51 a la 42:

```
ORDENADOR:  1       PERSONA:  1

      12345678
      --------
    8  . . . O 8
    7 . . O .  7
    6  . O . O 6
    5 . O . P  5
    4  . O P P 4
    3 P . P P  3
    2  P . P . 2
    1 . . . .  1
      --------
      12345678

MUEVO DESDE LA CASILLA?  37
             A LA CASILLA?  48
```

Unas cuantas jugadas después, el programa tiene ante sí un montón de posibilidades:



```
ORDENADOR: 1        PERSONA: 1

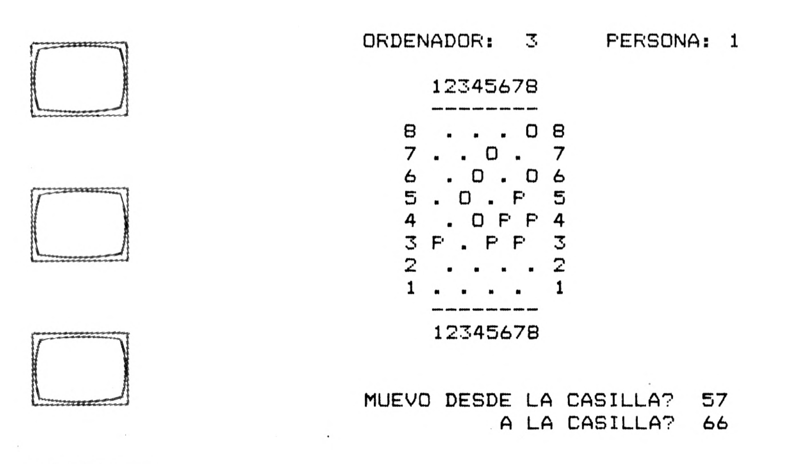
  12345678
  --------
8  . . . O 8
7 . . O .  7
6  . O . O 6
5 . O . P  5
4  . O P P 4
3 P O P P  3
2  P . . . 2
1 . . . .  1
  --------
  12345678

MOVIENDO DESDE LA CASILLA 44 A LA 26 SE CAPTURA EN LA CASILLA 55
ESTOY CONSIDERANDO MOVER DESDE LA CASILLA  44  A LA  53
MOVIENDO DESDE LA CASILLA 33 A LA 11 SE CAPTURA EN LA CASILLA 22

         >> CAPTURA REALIZADA
HE CAPTURADO Y LLEGADO EN LA CASILLA  11 A LA ULTIMA FILA
```

RISITAS ha escogido la mejor jugada posible dada la situación del tablero: comer una ficha al adversario y terminar al mismo tiempo en la última hilera del lado opuesto del tablero (en la 11); gana así dos puntos (y desaparece posteriormente del tablero).

Las posibilidades con que cuenta el jugador humano no son muy sabrosas que digamos. La ficha situada en la casilla 35 se encuentra atacada por la ficha del ordenador situada en la 44 y no parece que haya modo alguno de evitar ese peligro, ya que ni puede moverla ni puede comer con ella la ficha del ordenador. El jugador humano ignora esta amenaza, ya que nada puede hacer por salvar la ficha, y se esfuerza por aproximarse al lado opuesto del tablero con alguna de sus fichas (amenazando, ligeramente sólo, porque esta misma ficha carece de protección, la ficha del ordenador que ocupa la casilla 75):

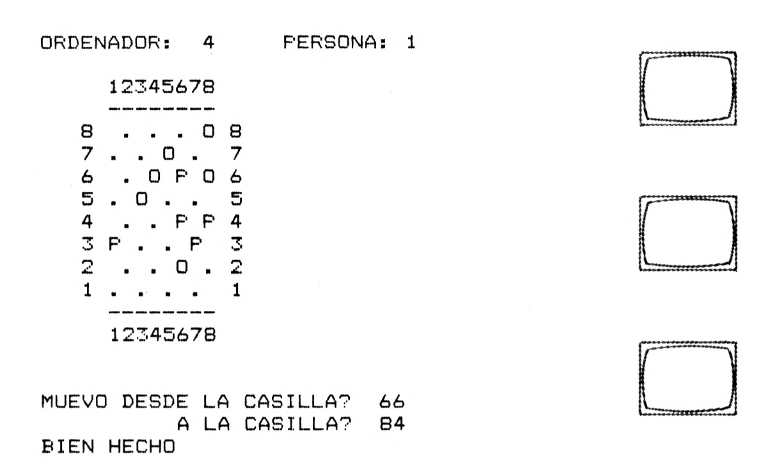


```
ORDENADOR: 3        PERSONA: 1

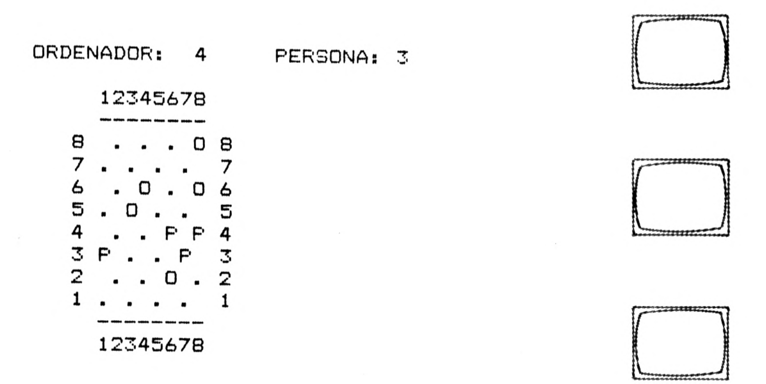
  12345678
  --------
8  . . . O 8
7 . . O .  7
6  . O . O 6
5 . O . P  5
4  . O P P 4
3 P . P P  3
2  . . . . 2
1 . . . .  1
  --------
  12345678

MUEVO DESDE LA CASILLA?  57
          A LA CASILLA?  66
```

RISITAS decide no comer la ficha situada en la casilla 65 para hacerlo con la de la casilla 35, colocándose así en la 26. El jugador humano le anuncia entonces que va a comer la ficha de la casilla 75, y el ordenador le felicita por ello:



El jugador humano ha ganado dos puntos con este salto, ya que ha terminado en la última hilera del lado opuesto del tablero. El ordenador lleva cuatro puntos (le falta uno para ganar) y el humano, tres:



Una de las prioridades dentro de la jerarquía de jugadas establecidas por RISITAS es la de mover a la última hilera del lado opuesto al suyo siempre que tenga ocasión de hacerlo.

ESTOY MOVIENDO HACIA LA ULTIMA FILA DESDE 26

Esta jugada, por supuesto, le otorga la victoria al ordenador:

```
ORDENADOR:  5     PERSONA: 3

    12345678
    --------
  8  . . . O 8
  7 . . . .  7
  6  . O . O 6
  5 . O . .  5
  4  . . P P 4
  3 P . . P  3
  2  . . . . 2
  1 . . . .  1
    --------
    12345678

EL JUEGO SE HA TERMINADO

YO SOY EL VENCEDOR
```

Cómo funciona el programa

Al igual que otros programas de este libro, RISITAS está construido en torno a un bucle principal que se ejecuta una y otra vez hasta satisfacer una condición determinada. Dentro de ese bucle hay un número de llamadas a las diferentes subrutinas.

```
 10 REM RISITAS
 20 GOSUB 2070: REM INICIALIZACION
 30 GOSUB 1760: REM IMPRESION DEL TABLERO
 40 REM ** EL CICLO PRINCIPAL COMIENZA **
 50 GOSUB 190: REM MOVIMIENTO DEL ORDENADOR
 60 GOSUB 1760: REM IMPRESION DEL TABLERO
 70 IF MO > 4 THEN 120
 80 GOSUB 1950: REM ACEPTA EL MOVIMIENTO DE LA
    PERSONA
 90 GOSUB 1760: REM IMPRESION DEL TABLERO
100 IF MP < 5 THEN 50
110 REM ***************
```

Como puede verse si se observa el bucle principal, es precisamente éste el que facilita la tarea de comprender cómo está construido el programa. Asimismo, simplifica la tarea de encontrar los errores. Por ejemplo, si el programa no imprimiera el tablero correctamente, lo más sensato sería mirar primero la subrutina que empieza en la línea 1760, la rutina llamada IMPRESION DEL TABLERO.

La acción pasa primero por la rutina de INICIACION, a partir de la línea 2070:

```
2070 REM INICIALIZACION
2080 DEFINT A - Z
2090 CLS
2100 RANDOMIZE
2110 DIM A(110): REM TABLERO Y ESPACIOS ALREDEDOR
     Y DETRAS DE LA PIEZA
2120 DIM G(3): REM ALMACENA UNA BUENA CAPTURA SE
     GURA
2130 DIM S(3): REM ALMACENA UNA CAPTURA SEGURA
2140 DIM T(18): REM ALMACENA OTRAS CAPTURAS. TAM
     BIEN SE USA PARA ALMACENAR MOVIMIENTOS DE
     NO-CAPTURA
2150 E = ASC(" "): REM CASILLA 'BLANCA' VACIA
2160 B = ASC("."): REM PCASILLA 'NEGRA' VACIA
2170 O = ASC("O"): REM PIEZA DEL ORDENADOR
2180 P = ASC("P"): REM PIEZA DE LA PERSONA
2190 PM = 0: REM MARCADOR DE LA PERSONA
2200 OM = 0: REM MARCADOR DEL ORDENADOR
2210 REM ** INICIALIZACION DEL TABLERO **
2220 FOR J = 10 TO 80 STEP 10
2230 FOR K = 1 TO 8
2240 READ X: A(J + K) = X
2250 NEXT K
2260 NEXT J
2270 RETURN
2280 REM ********************************
2290 DATA 80, 32, 80, 32, 80, 32, 80, 32
2300 DATA 32, 80, 32, 80, 32, 80, 32, 80
2310 DATA 46, 32, 46, 32, 46, 32, 46, 32
2320 DATA 32, 46, 32, 46, 32, 46, 32, 46
2330 DATA 46, 32, 46, 32, 46, 32, 46, 32
2340 DATA 32, 46, 32, 46, 32, 46, 32, 46
2350 DATA 79, 32, 79, 32, 79, 32, 79, 32
2360 DATA 32, 79, 32, 79, 32, 79, 32, 79
```

Aquí se dimensionan varias matrices. Son las siguientes:

A - para contener el tablero y las casillas "fuera del tablero" que le rodean;
G - que actúa de almacén para las jugadas clasificadas como "buena captura, segura" que encuentre durante un peinado del tablero:
S - tiene la misma función que G, pero las jugadas que almacena son menos deseables, las definidas como "seguras";
T - esta matriz contiene las jugadas o capturas que no pueden incluirse en ninguna de las dos clasificaciones anteriores.

Las sentencias REM identifican las variables que se asignan aquí, representando E una casilla blanca vacía, B una casilla negra vacía (que en la pantalla aparece como un punto), O las fichas del ordenador y P las fichas del jugador humano. Parece adecuado

utilizar nombres variables que nos recuerden lo que éstas representan. MP contiene los puntos que va marcando el jugador humano y MO, los del ordenador.

Las líneas 2210 a 2260 leen la configuración inicial del tablero en la matriz A.

El ciclo principal nos avisa de cómo se comportará el ordenador a partir de este punto. No nos detendremos a observar cómo se imprime el tablero ni cómo se aceptan las jugadas del jugador humano porque no nos parece que éstos sean problemas muy importantes en la programación.

Cuando el programa piensa sus jugadas sigue —como ya señalábamos anteriormente— un orden jerárquico estricto. Se fijan tres variables a cero con las líneas 220, 230 y 240; estas variables se utilizan cada vez que se ejecuta el programa. Las sentencias REM las explican:

```
180 REM ************************
190 REM MOVIMIENTO DEL ORDENADOR
200 REM ************************
210 REM BUSQUEDA PARA REALIZAR CAPTURAS
220 GSEGURA = 0: REM PARA CONTABILIZAR LAS CAPTU
    RAS BUENAS DE LAS PIEZAS AMENAZADAS DE LA PER
    SONA
230 CSEGURA = 0: REM PARA CONTABILIZAR LAS CAPTU
    RAS SEGURAS QUE NO ESTAN AMENAZADAS POR EL OR
    DENADOR
240 CAPTURA = 0: REM PARA CONTABILIZAR OTRAS CAP
    TURAS HECHAS
```

Los "almacenes" están vacíos:

```
250 FOR J = 1 TO 3
260 G(J) = 0: REM EL ALMACENAMIENTO DE CAPTURAS
    BUENAS ESTA VACIO
270 S(J) = 0: REM EL ALMACENAMIENTO DE CAPTURAS
    SEGURAS ESTA VACIO
280 T(J) = 0: REM EL ALMACENAMIENTO DE OTRAS CAP
    TURAS ESTA VACIO
290 NEXT J
```

Empieza ahora el ordenador el primer peinado del tablero, omitiendo el proceso de evaluación (véase la línea 320) si la casilla que está considerando no tiene una ficha suya. Merece la pena seguir en detalle la secuencia completa de la captura. Las sentencias REM explican el código casi completamente:

```
300 FOR J = 80 TO 30 STEP -10
310 FOR K = 1 TO 8
320 IF A(J + K) <> 0 THEN 390: REM EVITA LOS CAL
    CULOS EN EL CASO DE QUE NO HAYA UNA PIEZA DEL
    ORDENADOR
330 REM ** CAPTURA HACIA EL LADO DERECHO **
340 X = J + K - 9: Y = J + K - 18: Z = J + K - 27
    : M = -11
350 IF A(X) = P AND A(Y) = B THEN GOSUB 700: REM
    CAPTURA ENCONTRADA
360 REM ** CAPTURA HACIA EL LADO IZQUIERDO **
370 X = J + K - 11: Y = J + K - 22: Z = J + K -
    33: M = -9
380 IF A(X) = H AND A(Y) = B THEN GOSUB 700: REM
       CAPTURA ENCONTRADA
390 NEXT K
400 NEXT J
410 IF GSEGURA + CSEGURA + CAPTURA = 0 THEN 980:
    REM NO SE HA ENCONTRADO CAPTURA
420 REM ** AHORA SE ELIGE LA CAPTURA QUE SE VA A
        HACER **
430 PRINT: PRINT TAB(8) ">> CAPTURA REALIZADA"
440 FOR T = 1 TO 1000: NEXT T
450 IF GSEGURA <> 0 THEN 500
460 IF CSEGURA <> 0 THEN 670
470 REM ** AHORA SE ELIGE DE TODAS LAS POSIBLES
        CAPTURAS **
480 MOVIMIENTO = T(INT(RND(1) * CAPTURA) + 1)
490 GOTO 540
500 REM **  SE ELIGE UNA BUENA CAPTURA **
510 REM ** SELECCION DE LOS MOVIMIENTOS ALMACENA
        DOS **
520 MOVIMIENTO = G(INT(RND(1) * GSEGURA) + 1)
530 REM ** REALIZA UN MOVIMIENTO **
540 INICIO = INT(MOVIMIENTO/100)
550 ED = MOVIMIENTO - 100 * INICIO
560 A(INICIO) = B
570 A(INICIO - ED) = B
580 A(INICIO - 2 * ED) = O
590 MO = MO + 1
600 REM ** COMPROBACION DE PUNTOS EXTRAS, CASO DE
        HABER LLEGADO A LA ULTIMA FILA "
610 IF INICIO - 2 * ED > 18 THEN RETURN
620 A(INICIO - 2 * ED) = B
630 MO = MO + 1
640 PRINT "HE CAPTURADO Y LLEGADO EN LA CASILLA "
         ; INICIO - 2 * ED; "A LA ULTIMA FILA"
650 FOR T = 1 TO 2000: NEXT T
660 RETURN
670 REM ** CAPTURA SEGURA **
680 MOVIMIENTO = S(INT(RND(1) * CSEGURA) + 1)
690 GOTO 540
700 REM ** SE VERIFICA LA CAPTURA PROPUESTA POR
        MOTIVOS DE SEGURIDAD **
710 REM ** SE COMPRUEBA LA CASILLA INFERIOR EN LA
```

```
        MISMA DIRECCION EN QUE SE PRETENDE MOVER
        **
720 PRINT "MOVIENDO DESDE LA CASILLA "; J + K;
         "A LA "; Y; "SE CAPTURA EN LA CASILLA "
         ; X
730 FOR T = 1 TO 900: NEXT T
740 IF A(Z) = P THEN 920: REM SE ALMACENA COMO
    UNA CAPTURA NO SEGURA
750 REM ** SE VERIFICA LA CASILLA EN OTRA DIREC
        CION DE LA QUE SE PRETENDE MOVER"
760 IF A(Y + M) = P AND A(Y - M) = B THEN 920
770 REM ** AHORA SE VERIFICA SI EL MOVIMIENTO DEJA
        RIA LA PIEZA EXPUESTA A LA CAPTURA DE LA
        PERSONA **
780 IF A(J + K + M) = O AND A(J + K + 2 * M) = P
    THEN 920
790 REM ** SI SE ALCANZA ESTE PUNTO ENTONCES LA
        CAPTURA ES 'SEGURA' **
800 REM ** ALMACENA ESTE MOVIMIENTO **
810 CSEGURA = CSEGURA + 1
820 S(CSEGURA) = 100 * (J + K) + 20 + M: REM
830 REM ** AHORA VEREMOS SI ESTO ES REALMENTE UNA
        CAPTURA 'BUENA' O 'SEGURA'
840 VERIFICAR = GSEGURA
850 IF Y + 2 * M < 1 THEN RETURN
860 IF A(Y + M) = P AND A(Y - (20 + M)) <> B AND
    A(Y + 2 * M) = B THEN GSEGURA = GSEGURA + 1
870 IF VERIFICAR = GSEGURA THEN RETURN : REM **
    ESTE MOVIMIENTO NO ES UN 'BUEN MOVIMIENTO' **
880 REM ** ALMACENAMIENTO DE UN BUEN MOVIMIENTO
        SEGURO **
890 PRINT "ESTOY CONSIDERANDO MOVER DESDE LA CASI
         LLA "; J + K; "A LA "; M + 20 + J + K
900 G(GSEGURA) = 100 * (J + K) + 20 + M
910 RETURN
920 REM ** ALMACENA UNA CAPTURA NO SEGURA **
930 CAPTURA = CAPTURA + 1
940 PRINT "ESTOY CONSIDERANDO MOVER DESDE LA CASI
          LLA "; J + K; "A LA "; M + 20 + J + K
950 T(CAPTURA) = 100 * (J + K) + 20 + M
960 RETURN
```

Se ha de observar que la jugada propuesta se almacena en la línea 820 como un solo número. El resultado de esta manipulación es un número de cuatro cifras en el cual las dos primeras representan la casilla "desde la que se mueve" (o INICIO, como se la llama en varias partes del programa) y las otras dos cifras representan la casilla "a la que se va a llegar" (llamada ED en el programa; END hubiera producido colisiones en muchos sistemas, si fuera una palabra reservada de BASIC).

La rutina 1510 descodifica el número de cuatro cifras y hace la jugada:

```
1510 INICIO = INT(MOVIMIENTO/100)
1520 ED = MOVIMIENTO - 100 * INICIO
1530 A(INICIO) = B
1540 A(ED) = 0
1550 RETURN
```

Si el programa ha encontrado una jugada "buena captura, segura" (o más de una), la hace y después deja que mueva el jugador humano. Si no ha encontrado una jugada "buena captura, segura", pero tiene una de las clasificadas como "segura", hace ésa. En el caso de no contar con ninguna de estas dos, hará una de las clasificadas simplemente como "captura".

De fallar todas ellas, el programa pasa al siguiente elemento de su orden jerárquico de jugadas y mueve para proteger la ficha que encuentre amenazada por el jugador humano.

```
 980 REM ** MOVIMIENTO PARA PROTEGER LA PIEZA BAJO
         AMENAZA **
 990 MOVIMIENTO = 0
1000 J = 80
1010 K = 1
1020 Q = J + K
1030 IF A(Q) <> O THEN 1110: REM NO SE CONDIDERA
     ESTA CASILLA, PORQUE NO HAY EN ELLA NINGUNA
     PIEZA DEL ORDENADOR
1040 IF A(Q + 9) = B AND A(Q - 9) = P AND A(Q +
     18) = O THEN MOVIMIENTO = 100 * (Q + 18) +
     Q + 9
1050 REM ** ELIMINACION ALEATORIA DE ESTE MOVIMIEN
         TO SI SE EXPONE A UNO SUBSIGUIENTE **
1060 IF MOVIMIENTO <> 0 AND A(Q - 2) = P AND A(Q +
     20) = B AND RND(1) > .5 THEN 1510
1070 IF A(Q + 9) = B AND A(Q - 9) = P AND A(Q +
       20) = O THEN MOVIMIENTO = 100 * (Q + 20) +
       Q + 9: GOTO 1510
1080 IF A(Q + 11) = B AND A(Q - 11) = P AND A(Q +
     22) = O THEN MOVIMIENTO = 100 * (Q + 22) + Q
     + 11
1090 IF MOVIMIENTO <> 0 AND A(Q + 2) = P AND A(Q +
     22) = B AND RND(1) > .5 THEN 1510
1100 IF A(Q + 11) = B AND A(Q - 11) = P AND A(Q +
     20) = O THEN MOVIMIENTO = 100 * (Q + 20) + Q
     + 11: GOTO 1510
1110 IF K < 8 THEN K = K + 1: GOTO 1020
1120 IF J > 10 THEN J = J - 10: GOTO 1010
```

Si la línea 1040 encuentra tal jugada, la siguiente línea comprobará que ésta no pone en peligro ninguna otra ficha. De hacerlo, la jugada propuesta será rechazada en un 50 por 100 de los casos. Este no es un mecanismo de elección muy sofisticado que digamos, pero garantiza que el ordenador no siempre moverá a ciegas cuando se trata de proteger una de sus fichas (una ceguera que el jugador humano podría descubrir y explotar a su favor) y contribuye, además, a que las partidas no siempre sean iguales.

El conseguir poner una ficha en la última hilera del lado opuesto del tablero puntúa igual que el comer una ficha del adversario; así pues, el siguiente elemento en el orden jerárquico de jugadas es mover a una casilla de esta hilera, siempre que sea posible. La rutina que empieza en la línea 1140 se encarga de ello. Antes explicaba cómo la secuencia de casillas comprobadas en esta sección tenía por objeto el que las fichas situadas en las casillas centrales entraran en el santuario antes que aquellas situadas en las de los extremos:

```
1140 REM ** NO SE ENCUENTRA CAPTURA. SE INICIA BUS
        QUEDA PARA 'DESAPARECER' EN LA ULTIMA FI
        LA **
1150 MOVIMIENTO = 0
1160 REM PRIMERO SE HAN COMPROBADO MOVIMIENTOS NO
        ACONSEJABLES, ASI QUE PUEDEN SER REEMPLA
        ZADOS POR OTROS MEJORES
1170 IF A(22)= O AND A(11)= B THEN MOVIMIENTO = 22
1180 IF A(28)= O AND A(17)= B THEN MOVIMIENTO = 28
1190 IF A(22)= O AND A(13)= B THEN MOVIMIENTO = 22
1200 IF A(26)= O AND A(17)= B THEN MOVIMIENTO = 26
1210 IF A(26)= O AND A(15)= B THEN MOVIMIENTO = 26
1220 IF A(24)= O AND A(15)= B THEN MOVIMIENTO = 24
1230 IF A(24)= O AND A(13)= B THEN MOVIMIENTO = 24
1240 IF MOVIMIENTO = 0 THEN 1310
1250 PRINT: PRINT "ESTOY MOVIENDO HACIA LA ULTIMA
                FILA DESDE"; MOVIMIENTO
1260 FOR T = 1 TO 2000: NEXT T
1270 A(MOVIMIENTO) = B
1280 MO = MO + 1
1290 RETURN
```

Si esto no fuera posible, el programa peina el tablero para encontrar una jugada correcta y que no le ponga en peligro. La variable MOVIMIENTO cuenta las jugadas y la línea 1500 se encarga de escoger la jugada que se va a realizar:

```
1310 REM ** ALMACENA MOVIMIENTOS DE NO-CAPTURA **
1320 CMOVIMIENTO = 0: REM
1330 FOR J = 80 TO 30 STEP -10
1340 FOR K = 1 TO 8
1350 IF A(J + K) <> O THEN 1460
```

```
1360 X = J + K - 9: Y = J + K - 18: Z = J + K - 20
1370 Q = J + K + 2
1380 IF A(X) <> B THEN 1460
1390 IF A(Y)= P OR A(Z) = P AND A(Q) = B THEN 1460
1400 GOSUB 1560
1410 X = J + K -11: Y = J + K - 22: Z = J + K - 20
1420 Q = J + K - 2
1430 IF A(X) <> B THEN 1460
1440 IF A(Y)= P OR A(Z) = P AND A(Q) = B THEN 1460
1450 GOSUB 1560
1460 NEXT K
1470 NEXT J
1480 IF CMOVIMIENTO = 0 THEN 1630
1490 REM ** REALIZA EL MOVIMIENTO **
1500 MOVIMIENTO = T(INT(RND(1) * CMOVIMIENTO) + 1)
```

Si todavía no ha conseguido encontrar una jugada, RISITAS intenta encontrar cualquier jugada que se atenga a las reglas del juego. Escoge al azar entre 200 jugadas (contándolas con la variable L) y si tampoco así la encuentra, se da por vencido con la línea 1710:

```
1630 REM MOVIMIENTO ALEATORIO DE NO-CAPTURA
1640 PRINT "BUSCO ALEATORIAMENTE UN MOVIMIENTO
            LEGAL"
1650 L = 0
1660 L = L + 1
1670 J = 10 * INT(RND(1) * 8 + 1)
1680 K = INT(RND(1) * 8 + 1)
1690 IF A(J + K) = 0 THEN 1720
1700 IF L < 200 THEN 1660
1710 PRINT: PRINT "TE CONCEDO EL JUEGO": END
1720 IF A(J + K - 9) = B THEN MOVIMIENTO = 100 *
     (J + K) + J + K - 9: GOTO 1510
1730 IF A(J + K - 11) = B THEN MOVIMIENTO = 100 *
     (J + K) + J + K - 11: GOTO 1510
1740 GOTO 1700
```

Aquí tenemos el listado completo de RISITAS (para usuarios del ZX Spectrum, véase Apéndices):

```
10 REM RISITAS
20 GOSUB 2070: REM INICIALIZACION
30 GOSUB 1760: REM IMPRESION DEL TABLERO
40 REM ** EL CICLO PRINCIPAL COMIENZA **
50 GOSUB 190: REM MOVIMIENTO DEL ORDENADOR
60 GOSUB 1760: REM IMPRESION DEL TABLERO
70 IF MO > 4 THEN 120
80 GOSUB 1950: REM ACEPTA EL MOVIMIENTO DE LA
   PERSONA
```

```
 90 GOSUB 1760: REM IMPRESION DEL TABLERO
100 IF MP < 5 THEN 50
110 REM ***************
120 REM FINAL DEL JUEGO
130 PRINT: PRINT "EL JUEGO SE HA TERMINADO"
140 PRINT
150 IF MP > MO THEN PRINT "TU HAS GANADO"
160 IF MO > MP THEN PRINT "YO SOY EL VENCEDOR"
170 END
180 REM ************************
190 REM MOVIMIENTO DEL ORDENADOR
200 REM ************************
210 REM BUSQUEDA PARA REALIZAR CAPTURAS
220 GSEGURA = 0: REM PARA CONTABILIZAR LAS CAPTU
    RAS BUENAS DE LAS PIEZAS AMENAZADAS DE LA PER
    SONA
230 CSEGURA = 0: REM PARA CONTABILIZAR LAS CAPTU
    RAS SEGURAS QUE NO ESTAN AMENAZADAS POR EL OR
    DENADOR
240 CAPTURA = 0: REM PARA CONTABILIZAR OTRAS CAP
    TURAS HECHAS
250 FOR J = 1 TO 3
260 G(J) = 0: REM EL ALMACENAMIENTO DE CAPTURAS
    BUENAS ESTA VACIO
270 S(J) = 0: REM EL ALMACENAMIENTO DE CAPTURAS
    SEGURAS ESTA VACIO
280 T(J) = 0: REM EL ALMACENAMIENTO DE OTRAS CAP
    TURAS ESTA VACIO
290 NEXT J
300 FOR J = 80 TO 30 STEP -10
310 FOR K = 1 TO 8
320 IF A(J + K) <> O THEN 390: REM EVITA LOS CAL
    CULOS EN EL CASO DE QUE NO HAYA UNA PIEZA DEL
    ORDENADOR
330 REM ** CAPTURA HACIA EL LADO DERECHO **
340 X = J + K - 9: Y = J + K - 18: Z = J + K - 27
    : M = -11
350 IF A(X) = P AND A(Y) = B THEN GOSUB 700: REM
    CAPTURA ENCONTRADA
360 REM ** CAPTURA HACIA EL LADO IZQUIERDO **
370 X = J + K - 11: Y = J + K - 22: Z = J + K -
    33: M = -9
380 IF A(X) = H AND A(Y) = B THEN GOSUB 700: REM
       CAPTURA ENCONTRADA
390 NEXT K
400 NEXT J
410 IF GSEGURA + CSEGURA + CAPTURA = 0 THEN 980:
    REM NO SE HA ENCONTRADO CAPTURA
420 REM ** AHORA SE ELIGE LA CAPTURA QUE SE VA A
        HACER **
430 PRINT: PRINT TAB(8) ">> CAPTURA REALIZADA"
440 FOR T = 1 TO 1000: NEXT T
450 IF GSEGURA <> 0 THEN 500
460 IF CSEGURA <> 0 THEN 670
470 REM ** AHORA SE ELIGE DE TODAS LAS POSIBLES
        CAPTURAS **
480 MOVIMIENTO = T(INT(RND(1) * CAPTURA) + 1)
```

```
490 GOTO 540
500 REM **  SE ELIGE UNA BUENA CAPTURA **
510 REM ** SELECCION DE LOS MOVIMIENTOS ALMACENA
        DOS **
520 MOVIMIENTO = G(INT(RND(1) * GSEGURA) + 1)
530 REM ** REALIZA UN MOVIMIENTO **
540 INICIO = INT(MOVIMIENTO/100)
550 ED = MOVIMIENTO - 100 * INICIO
560 A(INICIO) = B
570 A(INICIO - ED) = B
580 A(INICIO - 2 * ED) = 0
590 MO = MO + 1
600 REM ** COMPROBACION DE PUNTOS EXTRAS, CASO DE
        HABER LLEGADO A LA ULTIMA FILA "
610 IF INICIO - 2 * ED > 18 THEN RETURN
620 A(INICIO - 2 * ED) = B
630 MO = MO + 1
640 PRINT "HE CAPTURADO Y LLEGADO EN LA CASILLA "
          ; INICIO - 2 * ED; "A LA ULTIMA FILA"
650 FOR T = 1 TO 2000: NEXT T
660 RETURN
670 REM ** CAPTURA SEGURA **
680 MOVIMIENTO = S(INT(RND(1) * CSEGURA) + 1)
690 GOTO 540
700 REM ** SE VERIFICA LA CAPTURA PROPUESTA POR
        MOTIVOS DE SEGURIDAD **
710 REM ** SE COMPRUEBA LA CASILLA INFERIOR EN LA
        MISMA DIRECCION EN QUE SE PRETENDE MOVER
        **
720 PRINT "MOVIENDO DESDE LA CASILLA "; J + K;
          "A LA "; Y; "SE CAPTURA EN LA CASILLA "
          ; X
730 FOR T = 1 TO 900: NEXT T
740 IF A(Z) = P THEN 920: REM SE ALMACENA COMO
    UNA CAPTURA NO SEGURA
750 REM ** SE VERIFICA LA CASILLA EN OTRA DIREC
        CION DE LA QUE SE PRETENDE MOVER"
760 IF A(Y + M) = P AND A(Y - M) = B THEN 920
770 REM ** AHORA SE VERIFICA SI EL MOVIMIENTO DEJA
        RIA LA PIEZA EXPUESTA A LA CAPTURA DE LA
        PERSONA **
780 IF A(J + K + M) = 0 AND A(J + K + 2 * M) = P
    THEN 920
790 REM ** SI SE ALCANZA ESTE PUNTO ENTONCES LA
        CAPTURA ES 'SEGURA' **
800 REM ** ALMACENA ESTE MOVIMIENTO **
810 CSEGURA = CSEGURA + 1
820 S(CSEGURA) = 100 * (J + K) + 20 + M: REM
830 REM ** AHORA VEREMOS SI ESTO ES REALMENTE UNA
        CAPTURA 'BUENA' O 'SEGURA'
840 VERIFICAR = GSEGURA
850 IF Y + 2 * M < 1 THEN RETURN
860 IF A(Y + M) = P AND A(Y - (20 + M)) <> B AND
    A(Y + 2 * M) = B THEN GSEGURA = GSEGURA + 1
870 IF VERIFICAR = GSEGURA THEN RETURN : REM **
    ESTE MOVIMIENTO NO ES UN 'BUEN MOVIMIENTO' **
```

```
880 REM ** ALMACENAMIENTO DE UN BUEN MOVIMIENTO
         SEGURO **
890 PRINT "ESTOY CONSIDERANDO MOVER DESDE LA CASI
          LLA "; J + K; "A LA "; M + 20 + J + K
900 G(GSEGURA) = 100 * (J + K) + 20 + M
910 RETURN
920 REM ** ALMACENA UNA CAPTURA NO SEGURA **
930 CAPTURA = CAPTURA + 1
940 PRINT "ESTOY CONSIDERANDO MOVER DESDE LA CASI
           LLA "; J + K; "A LA "; M + 20 + J + K
950 T(CAPTURA) = 100 * (J + K) + 20 + M
960 RETURN
970 REM ****************************************
980 REM ** MOVIMIENTO PARA PROTEGER LA PIEZA BAJO
         AMENAZA **
990 MOVIMIENTO = 0
1000 J = 80
1010 K = 1
1020 Q = J + K
1030 IF A(Q) <> O THEN 1110: REM NO SE CONDIDERA
     ESTA CASILLA, PORQUE NO HAY EN ELLA NINGUNA
     PIEZA DEL ORDENADOR
1040 IF A(Q + 9) = B AND A(Q - 9) = P AND A(Q +
     18) = 0 THEN MOVIMIENTO = 100 * (Q + 18) +
     Q + 9
1050 REM ** ELIMINACION ALEATORIA DE ESTE MOVIMIEN
         TO SI SE EXPONE A UNO SUBSIGUIENTE **
1060 IF MOVIMIENTO <> 0 AND A(Q - 2) = P AND A(Q +
     20) = B AND RND(1) > .5 THEN 1510
1070 IF A(Q + 9) = B AND A(Q - 9) = P AND A(Q +
       20) = 0 THEN MOVIMIENTO = 100 * (Q + 20) +
       Q + 9: GOTO 1510
1080 IF A(Q + 11) = B AND A(Q - 11) = P AND A(Q +
     22) = 0 THEN MOVIMIENTO = 100 * (Q + 22) + Q
     + 11
1090 IF MOVIMIENTO <> 0 AND A(Q + 2) = P AND A(Q +
     22) = B AND RND(1) > .5 THEN 1510
1100 IF A(Q + 11) = B AND A(Q - 11) = P AND A(Q +
     20) = 0 THEN MOVIMIENTO = 100 * (Q + 20) + Q
     + 11: GOTO 1510
1110 IF K < 8 THEN K = K + 1: GOTO 1020
1120 IF J > 10 THEN J = J - 10: GOTO 1010
1130 REM ****************************************
1140 REM ** NO SE ENCUENTRA CAPTURA. SE INICIA BUS
         QUEDA PARA 'DESAPARECER' EN LA ULTIMA FI
         LA **
1150 MOVIMIENTO = 0
1160 REM PRIMERO SE HAN COMPROBADO MOVIMIENTOS NO
         ACONSEJABLES, ASI QUE PUEDEN SER REEMPLA
         ZADOS POR OTROS MEJORES
1170 IF A(22)= 0 AND A(11)= B THEN MOVIMIENTO = 22
1180 IF A(28)= 0 AND A(17)= B THEN MOVIMIENTO = 28
1190 IF A(22)= 0 AND A(13)= B THEN MOVIMIENTO = 22
1200 IF A(26)= 0 AND A(17)= B THEN MOVIMIENTO = 26
1210 IF A(26)= 0 AND A(15)= B THEN MOVIMIENTO = 26
1220 IF A(24)= 0 AND A(15)= B THEN MOVIMIENTO = 24
1230 IF A(24)= 0 AND A(13)= B THEN MOVIMIENTO = 24
```

```
1240 IF MOVIMIENTO = O THEN 1310
1250 PRINT: PRINT "ESTOY MOVIENDO HACIA LA ULTIMA
                 FILA DESDE"; MOVIMIENTO
1260 FOR T = 1 TO 2000: NEXT T
1270 A(MOVIMIENTO) = B
1280 MO = MO + 1
1290 RETURN
1300 REM ***************************************
1310 REM ** ALMACENA MOVIMIENTOS DE NO-CAPTURA **
1320 CMOVIMIENTO = O: REM
1330 FOR J = 80 TO 30 STEP -10
1340 FOR K = 1 TO 8
1350 IF A(J + K) <> O THEN 1460
1360 X = J + K - 9: Y = J + K - 18: Z = J + K - 20
1370 Q = J + K + 2
1380 IF A(X) <> B THEN 1460
1390 IF A(Y)= P OR A(Z) = P AND A(Q) = B THEN 1460
1400 GOSUB 1560
1410 X = J + K -11: Y = J + K - 22: Z = J + K - 20
1420 Q = J + K - 2
1430 IF A(X) <> B THEN 1460
1440 IF A(Y)= P OR A(Z) = P AND A(Q) = B THEN 1460
1450 GOSUB 1560
1460 NEXT K
1470 NEXT J
1480 IF CMOVIMIENTO = O THEN 1630
1490 REM ** REALIZA EL MOVIMIENTO **
1500 MOVIMIENTO = T(INT(RND(1) * CMOVIMIENTO) + 1)
1510 INICIO = INT(MOVIMIENTO/100)
1520 ED = MOVIMIENTO - 100 * INICIO
1530 A(INICIO) = B
1540 A(ED) = O
1550 RETURN
1560 REM ** ALMACENA EL MOVIMIENTO **
1570 CMOVIMIENTO = CMOVIMIENTO + 1
1580 PRINT "ESTOY CONSIDERANDO MOVER DESDE LA CASI
            LLA "; J + K; "A LA "; X
1590 FOR T = 1 TO 800: NEXT T
1600 T(CMOVIMIENTO) = 100 * (J + K) + X
1610 RETURN
1620 REM ********************************
1630 REM MOVIMIENTO ALEATORIO DE NO-CAPTURA
1640 PRINT "BUSCO ALEATORIAMENTE UN MOVIMIENTO
            LEGAL"
1650 L = O
1660 L = L + 1
1670 J = 10 * INT(RND(1) * 8 + 1)
1680 K = INT(RND(1) * 8 + 1)
1690 IF A(J + K) = O THEN 1720
1700 IF L < 200 THEN 1660
1710 PRINT: PRINT "TE CONCEDO EL JUEGO": END
1720 IF A(J + K - 9) = B THEN MOVIMIENTO = 100 *
     (J + K) + J + K - 9: GOTO 1510
1730 IF A(J + K - 11) = B THEN MOVIMIENTO = 100 *
     (J + K) + J + K - 11: GOTO 1510
1740 GOTO 1700
1750 REM ********************
```

```
1760 REM IMPRESION DEL TABLERO
1770 CLS
1780 PRINT
1790 PRINT "ORDENADOR:"; MO;"    PERSONA:"; MP
1800 PRINT
1810 PRINT "       12345678"
1820 PRINT "       --------"
1830 FOR J = 80 TO 10 STEP - 10
1840 PRINT "    "; J/10;
1850 FOR K = 1 TO 8
1860 PRINT CHR$(A(J + K));
1870 NEXT K
1880 PRINT J/10
1890 NEXT J
1900 PRINT "       --------"
1910 PRINT "       12345678"
1920 PRINT
1930 RETURN
1940 REM **********************************
1950 REM ACEPTA EL MOVIMIENTO DE LA PERSONA
1960 INPUT "MUEVO DESDE LA CASILLA "; INICIO
1970 IF A(INICIO) < P THEN 1960
1980 INPUT "             A LA CASILLA "; ED
1990 IF A(ED) <> B OR ABS(INICIO - ED) > 11 AND
     A((INICIO + ED)/2) <> O THEN 19800 1510
2000 A(INICIO) = B
2010 A(ED) = P
2020 IF ABS(INICIO - ED) > 11 THEN A((INICIO + ED)
     / 2) = B: MP = MP + 1: PRINT "BIEN HECHO"
2030 IF ED > 80 THEN A(ED) = B: MP = MP + 1: PRINT
     "HAY UN PUNTO MAS PARA TI"
2040 FOR T = 1 TO 700: NEXT T
2050 RETURN
2060 REM **************
2070 REM INICIALIZACION
2080 DEFINT A - Z
2090 CLS
2100 RANDOMIZE
2110 DIM A(110): REM TABLERO Y ESPACIOS ALREDEDOR
     Y DETRAS DE LA PIEZA
2120 DIM G(3): REM ALMACENA UNA BUENA CAPTURA SE
     GURA
2130 DIM S(3): REM ALMACENA UNA CAPTURA SEGURA
2140 DIM T(18): REM ALMACENA OTRAS CAPTURAS. TAM
     BIEN SE USA PARA ALMACENAR MOVIMIENTOS DE
     NO-CAPTURA
2150 E = ASC(" "): REM CASILLA 'BLANCA' VACIA
2160 B = ASC("."): REM PCASILLA 'NEGRA' VACIA
2170 O = ASC("O"): REM PIEZA DEL ORDENADOR
2180 P = ASC("P"): REM PIEZA DE LA PERSONA
2190 PM = 0: REM MARCADOR DE LA PERSONA
2200 OM = 0: REM MARCADOR DEL ORDENADOR
2210 REM ** INICIALIZACION DEL TABLERO **
2220 FOR J = 10 TO 80 STEP 10
2230 FOR K = 1 TO 8
2240 READ X: A(J + K) = X
2250 NEXT K
```

```
2260 NEXT J
2270 RETURN
2280 REM ******************************
2290 DATA 80, 32, 80, 32, 80, 32, 80, 32
2300 DATA 32, 80, 32, 80, 32, 80, 32, 80
2310 DATA 46, 32, 46, 32, 46, 32, 46, 32
2320 DATA 32, 46, 32, 46, 32, 46, 32, 46
2330 DATA 46, 32, 46, 32, 46, 32, 46, 32
2340 DATA 32, 46, 32, 46, 32, 46, 32, 46
2350 DATA 79, 32, 79, 32, 79, 32, 79, 32
2360 DATA 32, 79, 32, 79, 32, 79, 32, 79
```



5
El valor de los juegos

Allá en los primeros momentos de la investigación de la inteligencia artificial se sugirió que la programación de juegos no era un trabajo que mereciera mucho la pena. El argumento que se esgrimía era que el esfuerzo realizado en calcular los algoritmos para un programa de AJEDREZ, por ejemplo, podría aplicarse mejor en la investigación de mecanismos que demostraran teoremas matemáticos o en programas que modelaran la manera en que opera el cerebro humano (dentro de los límites de lo que se entendía por esto entonces).

Pero los mecanismos mediante los cuales el cerebro consigue encontrar una solución a un problema complejo —como el que presenta un tablero de ajedrez en medio de una partida— no ha dejado nunca de fascinar. Mucho antes de que existieran los ordenadores (tal como los entendemos ahora) ya hubo quienes se devanaron los sesos pensando en cómo se podría escribir un programa de AJEDREZ.

En 1949, Claude Shannon (cuyo trabajo comentábamos en la sección APRENDIZAJE Y RAZONAMIENTO de este libro) presentó una ponencia en un congreso realizado en Nueva York. El trabajo se llamaba "Programming a Computer for Playing Chess". El valor de éste trasciende con mucho su importancia histórica como primera obra publicada sobre el tema. Un gran número de los conceptos que Shannon estudiaba en este trabajo se siguen utilizando todavía hoy en los programas de AJEDREZ.

Y lo que aún es más importante, Shannon se dio cuenta de que si se podían solucionar los problemas que entraña la programación de un ordenador para que juegue al ajedrez, las ideas que se sacaran de ello podrían ser muy valiosas a la hora de ayudar a las máquinas a desarrollar su competencia en otros campos en los que se dieran problemas de parecida complejidad. Señaló incluso algunos de esos campos: el diseño de circuitos electrónicos, las situaciones complicadas de tipo telefónico, la traducción y los problemas de deducción lógica.

Aquellos que despreciaron su trabajo por el simple hecho de que éste trataba de la construcción de máquinas capaces de jugar, se equivocaron de medio a medio. Cualquier avance en un campo determinado de la inteligencia artificial es potencialmente una fuente de información aplicable en cualquier otro campo de ésta. En el capítulo que dedicábamos al APRENDIZAJE Y RAZONAMIENTO vimos el programa TICTAC. No es muy significativo, en sí mismo, el contar con un programa que aprende por sí solo a mejorar su juego del TRES EN RAYA. Pero la propia idea de aprendizaje sí que es muy importante.

Las complejidades del mundo real

Hay muchas situaciones en el mundo que son el resultado de un asombroso encadenamiento de factores. Son demasiados los factores que han llevado a la situación presente para que el hombre pueda comprenderla fácilmente. Si además la situación es cambiante (como lo son todas las situaciones del mundo real), la capacidad del hombre para mantenerse al ritmo de ésta, a fin de tomar las decisiones más apropiadas sobre lo que hacer en cada momento, es casi ridícula de puro limitada.

Aquí es donde los ordenadores programados para el juego pueden venir en su ayuda. La experiencia que se adquiere al escribir una función de evaluación para un programa de AJEDREZ (una función de evaluación mide la fuerza o la debilidad global de uno de los contrincantes en términos de un número determinado de factores, entre los que se incluyen el número de fichas sobre el tablero, su naturaleza y posición, las otras casillas que pueden atacar, etc.), podría aplicarse perfectamente para producir una función de evaluación que indicara cuáles son las mejores medidas que se pueden tomar para terminar con problemas tales como la contaminación o el vertido de residuos nucleares.

Pensemos en la situación que se produjo en Three Mile Island debido al mal funcionamiento de un reactor nuclear. El número de variables que habían de considerarse sobrepasó la capacidad de los operadores humanos, como más tarde señalaría el informe realizado por el Comité Malone:

> ...el operador se vio bombardeado hasta tal punto por las pantallas, los pilotos indicadores de peligro, las impresoras y demás mecanismos, que le fue imposible detectar cualquier condición errónea y mucho menos evaluar cómo se debería actuar para corregir esa condición...

Un experto en ordenadores que pudiera desglosar toda la entrada *(input)* para localizar las partes verdaderamente importantes e indicar la acción a seguir, hubiera tenido una importancia inestimable en esa situación.

Parece probable, pues, que la experiencia o competencia adquirida trabajando con programas tales como los de AJEDREZ, puede ser decisiva para el desarrollo de otras áreas de la inteligencia artificial.

Los avances realizados de este modo no siempre resultan como se habían pronosticado. Por ejemplo, se han escrito programas de ajedrez de dos tipos: unos intentan imitar la manera en que los humanos juegan al ajedrez; y otros sencillamente tratan de jugar lo mejor posible. Se descubrió que los programas que intentan actuar siguiendo los mode-

los de los jugadores humanos, considerados en conjunto, no juegan tan bien como las máquinas que actúan en función de sus propios intereses.

De aquí podemos sacar dos lecciones. Una es que intentar modelar en una máquina los esquemas del pensamiento humano quizá no sea el mejor camino a seguir para obtener los niveles más altos de ejecución de inteligencia artificial. La segunda es que del hecho de intentar producir un programa que actúe como un ser humano, podemos sacar una idea genuina del modo en que se comportan las mentes humanas.

Otros juegos, otras lecciones

El AJEDREZ no fue, por supuesto, el único juego que se tomó en consideración. Las DAMAS y el TRES EN RAYA, por ejemplo, fueron otros de los primeros candidatos en atraer la atención de los investigadores en aquellos primeros días de la investigación de la inteligencia artificial.

En el capítulo dedicado al APRENDIZAJE Y RAZONAMIENTO comentamos el trabajo de Arthur Samuel, quien desarrolló un programa de DAMAS que aprendía a medida que jugaba. Cuando empezó, Samuel no era consciente de los problemas que entrañaba la escritura de un programa de DAMAS. Sabemos que empezó a desarrollarlo en 1946, cuando dejó de trabajar en los laboratorios Bell y se convirtió en profesor de la Universidad de Illinois.

Decidió que la universidad necesitaba un ordenador, pero los 110.000 dólares que le ofrecía la administración de la universidad no eran suficientes para comprar uno. Samuel pensó entonces que la única manera de conseguir una máquina era utilizar ese dinero para construir una ellos mismos. Pensó que, si era capaz de hacer algo espectacular con la primera máquina que estaban planeando construir, una máquina pequeñita, estaría en situación de pedir una financiación gubernamental. Samuel nos dice que creyó que las DAMAS, al ser un juego sencillito, sería fácil de programar. Una vez que lo tuvieran escrito, lo podrían utilizar para vencer al campeón mundial del momento en un campeonato que iba a tener lugar por entonces en Kankakee, una ciudad próxima a la universidad y de la publicidad que esto les diera, podrían conseguir financiación por otras vías.

Enseguida se hizo evidente la magnitud de la tarea. Hacia el momento en que debía celebrarse el campeonato, no se había terminado siquiera el ordenador y mucho menos el programa.

Samuel pensó en las DAMAS porque sabía que otros grupos estaban trabajando con el ajedrez. Consideró que las DAMAS, comparado con el AJEDREZ, era un juego bastante trivial. Pero, como pudimos ver en el capítulo APRENDIZAJE Y RAZONAMIENTO, incluso la programación de un juego como el TRES EN RAYA entraña no pocas dificultades.

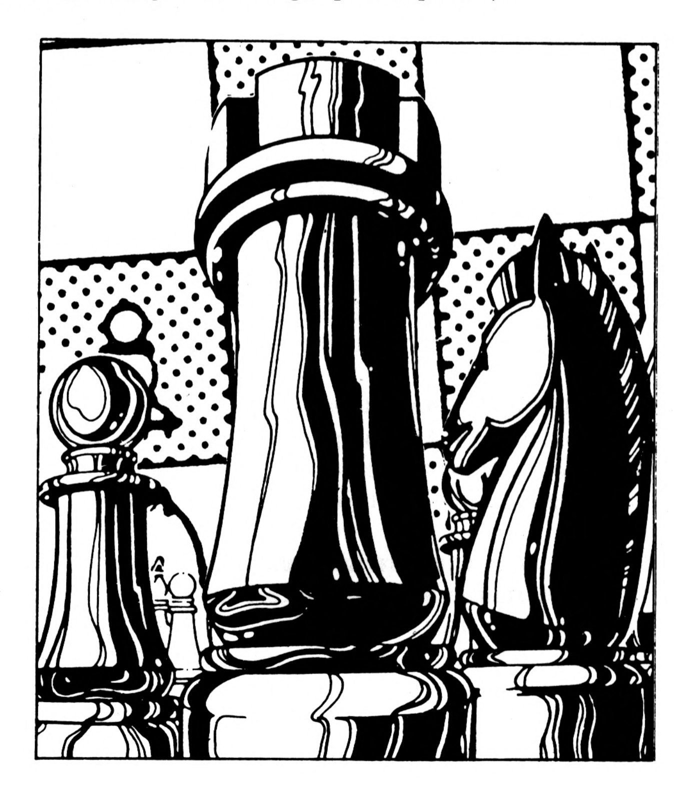
Si el TRES EN RAYA deja de ser trivial cuando se trata de programarlo, pensemos en lo que puede ser la programación de un juego como el GO. Mucho esfuerzo se ha aplicado a lo largo de la historia de la inteligencia artificial en la creación de programas de AJEDREZ, comparado con el relativamente escaso puesto en la programación del GO.

Hay tres razones que explican este hecho. Una es puramente cultural. La mayoría de nosotros, occidentales (el GO es un juego japonés), no jugamos normalmente al GO,

mientras que casi todos estamos, por lo menos, familiarizados con el AJEDREZ. La segunda razón es histórica. Los primeros investigadores en el campo de los juegos, como Turing y Shannon, resaltaron el ajedrez como un área que merecía la pena explorar. Y la tercera razón, señalada por J. A. Campbell (en su libro *'Go', his contribution to Computer Game-Playing, Theory and Practice,* editado por M. A. Bramer, Chichester, West Sussex: Ellis Horwood Ltd., 1983, p. 136), es que se ha demostrado que es extremadamente difícil escribir un programa que juegue siquiera al mismo nivel que un novato que acaba de aprender a jugar.

Pese a ello, es cierto que si uno quiere desafiar la dificultad que encierra el GO, quizá pueda hacerlo abordando sólo algunos aspectos del juego. En vez del tablero tradicional de 19 × 19, se podría escribir un programa para un tablero de siete por siete, por ejemplo, o limitar el número de jugadas posibles. Al igual que el OTHELLO se creó añadiendo una limitación a la jugada de apertura del REVERSI, se podría crear un tipo de "mini-GO" con un tablero pequeño y algunas restricciones en el juego.

Mientras que algunos juegos como el OTHELLO —en los que se pueden tabular bastante fácilmente los valores relativos de las diferentes casillas del tablero— responden bien a unas técnicas de investigación llevadas a cabo un poco por la "fuerza-bruta", se ha dicho que el GO sólo responderá a un tratamiento informático que utilice unas técnicas mucho más sutiles y depuradas. En realidad, es muy posible que el GO pase a ocupar el lugar reservado hasta ahora al AJEDREZ, considerándolo como la última prueba por la que ha de pasar la inteligencia artificial, por lo menos en lo que a los juegos se refiere (véase David Brown, Seeing is believing, *op cit.,* p. 177).





Sección tercera: EL HABLA

6
La comprensión del lenguaje natural

No cabe duda de que la capacidad de los ordenadores para comprender el lenguaje natural (es decir, el lenguaje comúnmente utilizado por los humanos para comunicarse) es una capacidad a partir de la que puede juzgarse, y se juzgará, la inteligencia o lo que quiera que sea que tienen los ordenadores.

La incapacidad del ordenador para conversar en nuestra lengua de cada día alza una barrera entre la máquina y nosotros mismos. Y tal barrera nos impide, por muchas ganas que tengamos de hacerlo, otorgarle al ordenador un grado mínimo de inteligencia.

Hay un par de programas que marcaron un hito en este campo. En esta sección del libro examinaremos unos programas que nos permitirán experimentar, aunque sólo sea un poco, el mismo entusiasmo que produjeron aquellos programas originales. Estos eran SHRDLU (la versión que aparece aquí se llama BLOQUELANDIA) y ELIZA (cuya ejecución en este libro, una de las más completas que se hayan publicado en lenguaje BASIC, se llama DOCTOR).

En el programa original de SHRDLU, un "robot" manipulaba bloques y otras formas coloreadas en respuesta a unas órdenes dadas en una lengua natural. Era capaz de llevar a cabo una soberbia conversación sobre lo que estaba haciendo en ese momento, lo que hizo en el pasado y por qué.

ELIZA, una imitación psiquiátrica que sigue las tendencias en este campo de Carl Rogers, causó tal efecto y asombro en un principio que su creador contaría posteriormente que había recibido angustiosas llamadas telefónicas de personas desesperadas que le pedían un poco más de tiempo de acceso al programa para ver si conseguían aclarar sus ideas.

Además de BLOQUELANDIA y DOCTOR, estudiaremos los problemas y potencial de las máquinas capaces de traducir. Se incluye en esta sección un programa bastante simple llamado TRADUCTOR y capaz de generar frases en *"spanglish"*, que ilustra el

tipo de soluciones que pueden dar los ordenadores cuando intentan manejar no una, sino dos, lenguas naturales.

HANSHAN, el último programa que veremos en esta sección, crea poemas al azar. El nivel de este programa, comparado con el del resto de los programas que componen este libro, es bastante bajo y además puede decirse que apenas ofrece una evidencia del poder cerebral del ordenador que lo ejecuta. Sin embargo, si el lector hubiera leído estas mismas líneas no hace más de treinta años, se hubiera quedado totalmente sorprendido de que un autor hablara como quien no quiere la cosa, sin darle mayor importancia, de una máquina capaz de escribir poesía y, lo que es más, de que tras esto, despachara el invento diciendo que carece de importancia. Hace treinta años hubiera sido un acontecimiento capaz de conmover al mundo. La proximidad a la maravilla ha embotado nuestra capacidad de percibirla y apreciarla.

No obstante, algunos de los resultados producidos por los programas incluidos en esta sección deberían maravillarnos aunque sólo fuera un poquito. Antes de empezar a comentar y ejecutar los programas, es necesario que estudiemos algunos de los problemas que impiden que la comunicación entre el hombre y la máquina en una lengua natural sea perfecta.

Análisis del lenguaje

Análisis es la palabra con que se define a la ruptura de las frases u oraciones en elementos que el ordenador puede manejar. El campo de la lingüística informática tradicionalmente se ha dedicado a la investigación de los modos de análisis de la oración, con el fin de revelar la función de las diferentes partes de ésta en relación con su sintaxis. Se hace esto en la esperanza de que la máquina, al analizarla, pueda aproximarse a la comprensión de la oración que se procesa.

Sin embargo, cada vez se da un interés mayor por buscar un significado en términos de la función de la oración dentro de un marco de referencia mucho más amplio (como lo hacemos, en términos de una experiencia y un conocimiento del medio anteriores, cuando tratamos de entender una frase). Por supuesto, aunque la investigación basada en la estructura sintáctica continúa, cada vez se hace sentir más el empuje hacia enfoques en los que se tiene en cuenta una "visión global del medio".

La razón de esto es bastante obvia. Queremos poder hablar con los ordenadores en nuestros propios términos, más que en aquellos que nos dictan los límites de un lenguaje metálico. Cuando hablamos sobre un campo del conocimiento que nos interesa con un amigo que comparte nuestro interés, partimos de la base de que tenemos una gran cantidad de conocimiento y un bagaje intelectual comunes. Así es como nos gustaría hablar a los ordenadores, pudiendo dar por supuesta la existencia de un conocimiento básico que sirva de marco para la comunicación.

Imaginemos que somos los directores de una compañía minera. Contamos con un ordenador programado para ayudarnos en la tarea de buscar los minerales preciosos (al menos existe un programa de este tipo, PROSPECCION), y nos gustaría poder hablar con él utilizando las mismas palabras y frases que usamos cuando hablamos del tema con nuestros colegas.

El esfuerzo se reduce a dar al ordenador una visión del mundo que le permita mane-

jar una entrada en lenguaje natural, usando para ello los conocimientos que tiene, a modo de plantilla con la que poder comprobar los posibles significados.

En este capítulo descubrirá el lector que las únicas demostraciones convincentes de la ‘‘comunicación mediante un lenguaje natural’’ que podemos dar son para ‘‘visiones del mundo’’ extremadamente restringidas. En BLOQUELANDIA, por ejemplo, el mundo consiste en un espacio de dos dimensiones dentro del cual el ordenador manipula cuatro bloques de color. Sin embargo, el rendimiento del ordenador dentro de ese limitado universo es sorprendente, aun cuando no llega a las vertiginosas alturas de SHRDLU, el programa en el que éste se inspira.

SHRDLU, por ejemplo, podía contestar a frases tales como BUSCA UN BLOQUE MAS ALTO QUE EL QUE TIENES AHORA Y METELO EN LA CAJA. Veremos cómo BLOQUELANDIA, aun habitando un universo más restringido que aquél, se muestra incapaz de tal hazaña. Sin embargo, como también veremos a su debido tiempo, BLOQUELANDIA no lo hace mal del todo.

Como ya hemos dicho, el programa tiene que manipular cuatro bloques coloreados. Nos dice dónde están, ya sea encontrando un bloque determinado o describiendo toda la escena, y los mueve de un lado a otro. En la ejecución que precede al listado del programa, el bloque verde está encima del bloque naranja. Le pedí al ordenador que pusiera el bloque rojo encima del bloque naranja. Esto significa que el programa, en primer lugar, ha de bajar el bloque verde y después localizar el bloque rojo para ponerlo encima del naranja. Esta es la salida del programa (con lo que nos dice el ordenador):

```
......
......
......
...V..
..RN.A


? PON EL BLOQUE ROJO ENCIMA DEL NARANJA

     COMPRENDO
AHORA MOVERE EL BLOQUE VERDE

ESTOY MOVIENDOLO A LA COLUMNA  5
AHORA ESTOY PONIENDO EL BLOQUE ROJO
ENCIMA DEL BLOQUE NARANJA


          ......
          ......
          ......
          ...R..
          ...NVA
```

Como descubrirá el lector si llega a ejecutar el programa, el hecho de comunicarse en castellano con el ordenador (un castellano limitado, admitámoslo, pero castellano a pesar de todo) produce un efecto mágico al igual que el hecho de que el ordenador siga nuestras instrucciones y nos responda también en castellano.

Cuando se empezó a investigar la inteligencia artificial, se dedicó mucho tiempo a sa-

ber si el programa entendía o no en realidad lo que estaba sucediendo. Los investigadores tenían conciencia de que incluso programas como SHRDLU o el ELIZA de Joseph Weinzenbaum (que estudiaremos en profundidad un poco más adelante), aunque daban la impresión, de lo más convincente, por otro lado, de que se comportaban inteligentemente, no les habían acercado en lo más mínimo a la inteligencia "real" (independientemente de lo que cada uno entienda por ello).

Este problema hoy ha dejado de importar. Actualmente no perdemos el tiempo preguntándonos si el robot soldador utilizado en las cadenas de producción de automóviles, "ve de verdad" lo que está haciendo o "se siente satisfecho" ante el trabajo bien hecho. Lo importante es que la cosa funciona. En muchas ocasiones, como tendremos ocasión de comprobar en esta sección dedicada al lenguaje, basta y sobra con que el ordenador pueda manejarlo de un modo efectivo, como si "realmente" entendiera lo que oye y dice.

Los programas "sistemas expertos" (de los que hablaremos en detalle en el capítulo que lleva este nombre) pueden hacer nuevos descubrimientos y ayudar a los seres humanos en la solución de problemas difíciles. El lado pragmático de la inteligencia artificial tiende actualmente a considerar que "es inteligencia lo que actúa como tal". Si el ordenador se comporta inteligentemente —incluso dentro de un dominio extremadamente limitado— demos por hecho que entiende lo que está sucediendo y continuemos investigando las cuestiones realmente importantes, como la manera de incrementar la aparente inteligencia de la bestia.

Problemas

Son muchos y fundamentales los problemas con los que han de luchar los investigadores de la inteligencia artificial en su intento de desentrañar los misterios que encierra el procesado del lenguaje natural. La enorme cantidad de palabras de que consta cualquier lengua humana y la asombrosa cantidad de maneras en que éstas pueden combinarse constituyen el escollo fundamental y más obvio. Muchas de las frases que forman las oraciones son ambiguas. Generalmente, basándonos en un conocimiento previo, podemos salvar la ambigüedad y llegar a su significado. La ambigüedad suele ser inherente al discurso hablado —quizá más de lo que lo es a la comunicación escrita— y además éste, el discurso hablado, a menudo es incompleto y carece de estructuración.

Cualquier tarea adicional que se le dé al ordenador aumenta su tiempo de procesado. Un sistema que implique una lengua natural no debe exigir tanto tiempo que el proceso resulte inútil en términos humanos. Si al ordenador le lleva una semana "entender" un párrafo, se nos quitarán las ganas de dedicar mucho tiempo a la investigación de su capacidad para comunicarse con nosotros.

Sintaxis y Semántica

Constituyen dos maneras de abordar el campo del análisis del lenguaje. No se excluyen la una a la otra. Se utilizan para solucionar los problemas subyacentes al lenguaje aun en el uso más común y cotidiano de éste. Incluso tratándose de una frase tan sencilla

como la que vamos a ver ahora, tardamos unos instantes en saber quién es la persona que está cansada:

EL HOMBRE QUE ESTA CON PEDRO DIJO QUE ESTABA CANSADO

Leída fuera de contexto, como lo acaba de hacer el lector ahora, no tenemos ninguna clave que nos indique quién es el que está cansado, aunque yo me inclino más bien a pensar que el "cansado" es el hombre y no Pedro.

Cualquier sistema de análisis de una lengua natural ha de poder tratar e intentar resolver este tipo de problemas. Margaret Boden (en su libro *Artificial Intelligence and Natural Man,* Hassocks, Sussex: Harvester Press, 1977, p. 112), dando muestras de un humor delicioso, denomina "Problema del Arzobispo" a la dificultad de asignar automáticamente un significado. El origen de esta denominación lo encontramos en *Alicia en el país de las Maravillas:*

> "Incluso Stigand, el patriótico arzobispo de Canterbury, pensó que esto era aconsejable."
> "¿Pensó qué?"
> "Pensó esto", contestó el ratón, un tanto malhumorado. "Supongo que sabes lo que significa 'esto'."
> "Sé de sobra lo que significa 'esto' cuando encuentro una cosa", dijo el pato. "Suele ser una rana o un gusano. Lo que te pregunto es qué es lo que pensó el arzobispo."

Examinemos ahora una frase y veamos cómo la dividiría un "analizador", antes de procesar cada palabra con el fin de aproximarse a la comprensión del escrito analizado. (Luego examinaremos la importante cuestión de cómo ha de definirse la "comprensión".)

Esta es la frase:

EL DELGADO ANCIANO ESTABA BAJO EL ROBLE FRONDOSO

Podemos examinar la frase sintácticamente (poniendo entre paréntesis cada elemento sintáctico de la estructura), como sigue:

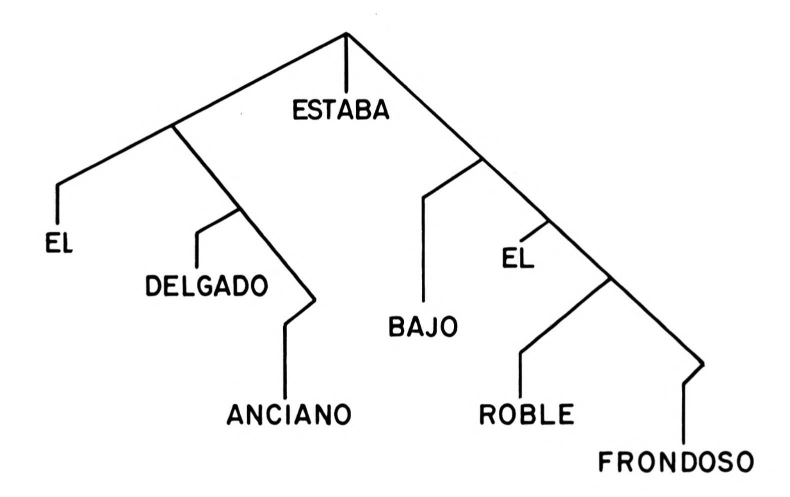
[[EL [[DELGADO] [ANCIANO]]]
ESTABA [[BAJO [EL [ROBLE] [FRONDOSO]]]]]

Observando con atención, siguiendo el sistema de paréntesis, uno puede hacerse una idea bastante razonable de cómo están ligados los diferentes elementos. Por ejemplo, DELGADO y ANCIANO llevan paréntesis individuales, [DELGADO] [ANCIANO], pero también están unidos mediante un paréntesis, formando un grupo más amplio, [[DELGADO] [ANCIANO]].

Hay un tercer paréntesis que encierra todo el lado izquierdo de la frase, [EL ... ANCIANO]]] y el verbo ESTABA sólo está ligado por los paréntesis que encierran la frase entera, ya que él mismo liga las dos partes de la frase. Consideremos ahora el lado de-

recho; vemos que BAJO está incluido en el mismo paréntesis que FRONDOSO, ya que un par de paréntesis unen el conjunto de este lado. EL sólo tiene paréntesis de un lado, lo que significa que su única función es modificar al sustantivo que le sigue (es evidente que 'el roble frondoso' es diferente de 'un roble frondoso'; el primero se refiere a un roble determinado).

Podemos expresar la estructura sintáctica de nuestra frase mediante un árbol:



Si consiguiéramos que el ordenador desmenuzara las frases como lo hemos hecho nosotros con ésta, que fuera capaz de reconocer las partes del discurso en cada rama del árbol y/o poner entre paréntesis cada elemento sintáctico, estaríamos en el camino de lograr cierto grado de comprensión.

Con esto volvemos a la cuestión que se planteaba hace un momento ¿Qué entendemos por "comprensión" en el contexto de la máquina? J. Klir y M. Valach (en su libro *Cybernetic Modelling,* Londres, Iliffe Books, 1965) sugieren que la comprensión de un mensaje hablado consta generalmente de tres partes:

1. Un modo de "oír el mensaje".
2. Un medio de responder a éste.
3. Un método para evaluar si la respuesta (2) estaba o no formulada de manera que muestre que se ha comprendido el mensaje.

Podrían considerarse varias maneras de evaluar la comprensión de un texto escrito, dice Geoff Simons (en su obra *Are Computers Alive?,* Brighton, Sussex, The Harvester Press, 1983, p. 129). Entre éstas se incluyen el suponer que ha habido comprensión por parte del ordenador cuando éste puede responder correctamente a las preguntas que le hagamos sobre el texto o el observar si la máquina es capaz de hacer conexiones inteligentes entre su propio conocimiento anterior (su base de datos) y la información que ha sacado de su "lectura" del texto.



7
BLOQUELANDIA

Algunas veces el vernos ante un ordenador que ''charla'' con nosotros en castellano nos resulta un poco inquietante. En BLOQUELANDIA, una versión simplificada del famoso programa llamado SHRDLU (que comentaré un poco más adelante), el ordenador manipula una serie de bloques de juguete de diferentes colores siguiendo las instrucciones que nosotros le damos en castellano y diciéndonos —de vez en cuando— cómo está colocado cada bloque en relación con los demás.

Los bloques, por supuesto, no existen como tales, ya que no son sino invenciones electrónicas del cerebro del ordenador. No obstante, vemos una representación de ellos en la pantalla y esta representación cambia según el ordenador va moviendo los bloques de un lado a otro.

Como el lector ya se habrá percatado, suele ser mucho más fácil obtener una demostración convincente de la inteligencia de la máquina cuando ésta opera en un dominio limitado. El dominio de los bloques o cubos de juguete se utiliza mucho en los experimentos de inteligencia artificial precisamente por eso, porque es un dominio claramente limitado, al tiempo que permite un grado considerable de interacción y manipulación, como enseguida tendrá el lector la ocasión de observar por sí mismo.

En el universo que el ordenador manipulará en este programa hay cuatro bloques. Estos son rojo (que aparece con la letra 'R'), verde ('V'), naranja ('N') y azul ('A'). Al inicio del programa esto es lo que vemos en la pantalla:



Lo que vemos es la parte delantera de los bloques. BLOQUELANDIA es esencialmente bidimensional. Aunque podemos mover los bloques y ponerlos unos encima de otros, no podemos situarlos delante o detrás de otro bloque. Los puntos son invisibles para el ordenador. Sólo existen para el usuario, para su propia comodidad y muestran los lugares que pueden ocupar los bloques.

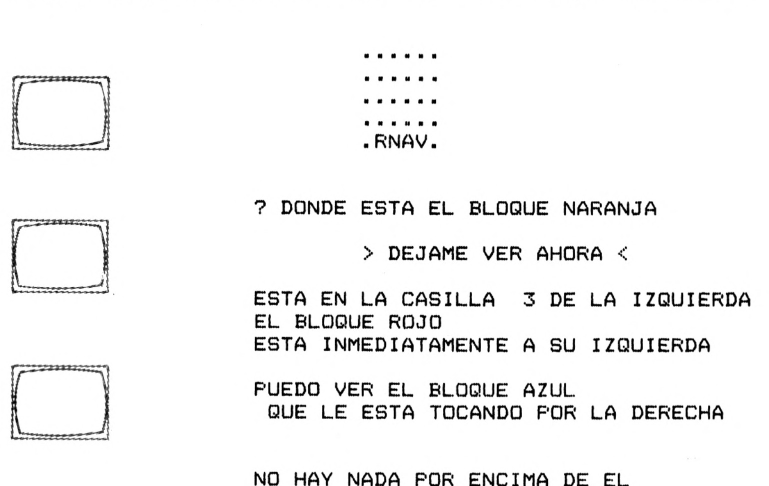
Pese a que el número de frases que podemos utilizar para comunicarnos con el ordenador es bastante restringido, estoy seguro de que todo aquel que lo ejecute se sentirá encantado al ver lo convincente que llega a resultar la conversación. Asimismo, constituye una demostración fácil de explicar (pero no por ello menos impresionante) del funcionamiento de la inteligencia artificial; lo más apropiado para esos amigos escépticos que todos tenemos.

Después de aparecer los bloques en la pantalla, vemos el signo recordatorio "?" lo que nos indica que BLOQUELANDIA está esperando que le digamos algo. Podemos pedirle que nos describa su mundo:

```
? DIME LO QUE PUEDES VER

EMPEZANDO DESDE LA DERECHA
...UN ESPACIO LUEGO
MIS SENSORES REGISTRAN EL BLOQUE VERDE
ESTA EL BLOQUE AZUL
MIS SENSORES REGISTRAN EL BLOQUE NARANJA
MIS SENSORES REGISTRAN EL BLOQUE ROJO
FINALMENTE ...UN ESPACIO
```

Una vez que nos ha dado esta visión de conjunto (cuando dice "espacio" se refiere a un punto), podemos pedirle que localice un bloque determinado dentro de este mundo:



Desde luego que este modo de informarnos de lo que hay no es nada del otro mundo, aun cuando, como observará el lector, el programa está respondiendo a preguntas hechas en una lengua natural y lo hace, además, en un castellano bastante aceptable. En-

tramos ahora en el corazón del asunto diciéndole al ordenador que manipule los elementos de su limitado dominio:

```
                ......
                ......
                ......
                ......
                .RNAV.


? PON EL BLOQUE VERDE ENCIMA DEL NARANJA

       DE ACUERDO
AHORA ESTOY PONIENDO EL BLOQUE VERDE
ENCIMA DEL BLOQUE NARANJA

        ......
        ......
        ......
        ..V...
        .RNA..


? DONDE ESTA EL NARANJA

        > DEJAME VER AHORA <

ESTA EN LA CASILLA  3 DE LA IZQUIERDA
EL BLOQUE ROJO
ESTA INMEDIATAMENTE A SU IZQUIERDA

PUEDO VER EL BLOQUE AZUL
 QUE LE ESTA TOCANDO POR LA DERECHA

ENCIMA ESTA EL BLOQUE VERDE

NO HAY NADA POR ENCIMA DE EL
```

Una cosa es poner un bloque (el "objeto") encima de otro (el "blanco") cuando este último no tiene nada encima, y otra muy distinta y que requiere además una cantidad significativa de código, es cuando primero se ha de dejar libre el blanco. La situación se complica todavía más cuando encima del bloque-objeto hay otro bloque (o más de uno) que ha de ser quitado antes de poder mover el primero. En esta ejecución, el programa tiene que dejar libre el bloque-blanco. El bloque-objeto no tiene ningún otro encima y por tanto no ofrece problema:

```
        ......
        ......
        ......
        ..V...
        .RNA..

? PON EL ROJO ENCIMA DEL NARANJA

     COMPRENDO
AHORA MOVERE EL BLOQUE VERDE

ESTOY MOVIENDOLO A LA COLUMNA  4
AHORA ESTOY PONIENDO EL BLOQUE ROJO
ENCIMA DEL BLOQUE NARANJA
```

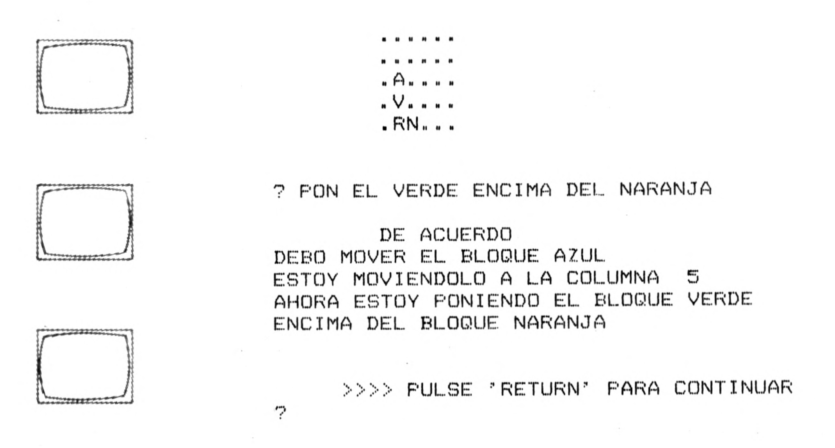
·Un poco más tarde, el orden de los bloques es el siguiente:

```
                ......
                ......
                .A....
                .V....
                .RN...


? DIME LO QUE VES

EMPEZANDO DESDE LA DERECHA
...UN ESPACIO LUEGO
...UN ESPACIO LUEGO
...UN ESPACIO LUEGO
MIS SENSORES REGISTRAN EL BLOQUE NARANJA
ESTA EL BLOQUE AZUL
...Y MAS ABAJO DE EL...
VEO EL BLOQUE VERDE
...Y MAS ABAJO DE EL...
VEO EL BLOQUE ROJO
FINALMENTE ...UN ESPACIO
```

Le pedimos ahora lo más difícil de todo lo que le hemos pedido hasta el momento:



Este programa se comporta de un modo bastante misterioso. Parece que entiende nuestras órdenes o, por lo menos, actúa de acuerdo con éstas. Se diría que comprende lo que está haciendo, ya que nos informa del resultado de sus actos.

El funcionamiento

¿Cómo lo consigue? ¿Cuál es el secreto de esta impresionante actuación? Todo el proyecto se reduce al análisis gramatical de las frases con el fin de llegar a comprender las partes importantes de las órdenes que se le dan, a la manipulación de los elementos en una matriz bidimensional en respuesta a las instrucciones recogidas en el análisis de la

entrada y, finalmente, a la impresión de la matriz de modo que exprese los contenidos con claridad para que puedan percibirse fácilmente.

Este tipo de "explicación", desde luego, no llega a quitarnos la impresión de inteligencia que nos da el programa; y eso que BLOQUELANDIA es una criatura de lo más primitiva si se la compara con su progenitor, SHRDLU.

Este programa, terminado en 1971 por Terry Winograd, no sólo manipula bloques, sino que los muestra en la pantalla, en color, creando su mundo al tiempo que actúa recíprocamente con éste. Oigamos cómo Terry habla con SHRDLU (la salida del ordenador son las mayúsculas precedidas por el signo >):

```
Recoge el bloque rojo grande
 > DE ACUERDO
Busca un bloque mas grande que el que
  tienes, e introducelo en la caja.
 > DE ACUERDO
Que hay en la caja?
 > LA PIRAMIDE AZUL Y LA CAJA AZUL
Quien sostiene a la piramide?
 > LA CAJA
Hay algun bloque grande detras de la piramide?
 > SI, HAY TRES: UNO ROJO GRANDE, UN CUBO
    VERDE GRANDE Y EL AZUL
```

Como puede verse, se trata de un esfuerzo notable y como tal lo consideran todos los investigadores, para quienes el trabajo de Winograd es una de las cotas más altas a las que se ha podido llegar en el desarrollo de programas que impliquen el uso de una lengua natural. A su lado, nuestro BLOQUELANDIA queda oscurecido.

No obstante, BLOQUELANDIA es capaz de una manipulación bastante complicada, aun cuando su salida no sea tan compleja como la que produce SHRDLU. Demostraré esta afirmación volviendo a nuestro programa y pidiéndole que haga para nosotros unas cuantas acrobacias.

En la versión castellana se ha modificado el programa, de modo que permita el uso conjunto de las palabras "BLOQUE" y "color", o la omisión de la primera, en todas sus posibles combinaciones, tanto a la hora de preguntar como de ordenar.

Esto agilizará la ejecución, aumentando al mismo tiempo las posibilidades del programa original, con la consiguiente sensación de mayor inteligencia.

Las distintas subrutinas que se encargan de ello son:

a) Líneas 1750 a 1820 (caso: '-color...-color').
b) Líneas 1830 a 1890 (casos: 'bloque-color...-color' y '-color...bloque-color').
c) Líneas 1530 a 1590 (caso: 'bloque-color...bloque-color').

A modo de ejemplo, las posibilidades de decir al ordenador que ponga el bloque azul encima del verde son las siguientes:

```
PON EL BLOQUE AZUL ENCIMA DEL BLOQUE VERDE

PON EL BLOQUE AZUL ENCIMA DEL VERDE

PON EL AZUL ENCIMA DEL BLOQUE VERDE

PON EL AZUL ENCIMA DEL VERDE


                  ......
                  ......
                  ......
                  ..A...
                  .RN.V.
```

Observemos que el bloque naranja está debajo del azul. Le pediremos ahora al programa que invierta la posición de ambos:

```
? PON EL NARANJA ENCIMA DEL AZUL

     COMPRENDO
DEBO MOVER EL BLOQUE AZUL
ESTOY MOVIENDOLO A LA COLUMNA  4
AHORA ESTOY PONIENDO EL BLOQUE NARANJA
ENCIMA DEL BLOQUE AZUL


      ......
      ......
      ......
      ...N..
      .R.AV.
```

Construiremos ahora una torre para ver cómo la manipula BLOQUELANDIA:

```
? PON EL BLOQUE VERDE ENCIMA DEL NARANJA

    COMPRENDO
AHORA ESTOY PONIENDO EL BLOQUE VERDE
ENCIMA DEL BLOQUE NARANJA


       ......
       ......
       ...V..
       ...N..
       .R.A..
```

Le decimos después al ordenador que queremos que ponga el bloque inferior de la torre sobre el bloque que está solo. Observemos que el programa está escrito de modo que garantiza que si el bloque-blanco está libre (no tiene ningún otro encima), no se le pondrá nada encima mientras se está dejando libre el bloque-objeto:

```
? PON EL AZUL ENCIMA DEL ROJO

        DE ACUERDO
DEBO MOVER EL BLOQUE VERDE
ESTOY MOVIENDOLO A LA COLUMNA  6
DEBO MOVER EL BLOQUE NARANJA
ESTOY MOVIENDOLO A LA COLUMNA  6
AHORA ESTOY PONIENDO EL BLOQUE AZUL
ENCIMA DEL BLOQUE ROJO


       ......
       ......
       ......
       .A...N
       .R...V
```

Ahora tenemos dos torrecitas, una con el bloque azul encima del rojo y otra con el bloque naranja encima del verde. Le pediremos al ordenador que manipule los dos bloques que están debajo. Como se ve, las pruebas que le exigimos a BLOQUELANDIA son cada vez más difíciles:

```
? PON EL VERDE ENCIMA DEL ROJO

        DE ACUERDO
DEBO MOVER EL BLOQUE NARANJA
ESTOY MOVIENDOLO A LA COLUMNA  5
DEBO MOVER EL BLOQUE AZUL
ESTOY MOVIENDOLO A LA COLUMNA  5
AHORA ESTOY PONIENDO EL BLOQUE VERDE
ENCIMA DEL BLOQUE ROJO


       ......
       ......
       ......
       .V..A.
       .R..N.
```

Lleva a cabo nuestra orden sin ningún problema. Usemos ahora el bloque rojo como objeto (en este momento está cubierto por el verde) y el azul como bloque-blanco para construir una torre con los cuatro:

```
? PON EL ROJO ENCIMA DEL AZUL

      COMPRENDO
DEBO MOVER EL BLOQUE VERDE
ESTOY MOVIENDOLO A LA COLUMNA  3
AHORA ESTOY PONIENDO EL BLOQUE ROJO
ENCIMA DEL BLOQUE AZUL


       ......
       ......
       ....R.
       ....A.
       ..V.N.
```

```
? PON EL VERDE ENCIMA DEL BLOQUE AZUL

     COMPRENDO
AHORA MOVERE EL BLOQUE ROJO

ESTOY MOVIENDOLO A LA COLUMNA  2
AHORA ESTOY PONIENDO EL BLOQUE VERDE
ENCIMA DEL BLOQUE AZUL


        ......
        ......
        ....V.
        ....A.
        .R..N.
```

```
? PON EL BLOQUE ROJO ENCIMA DEL BLOQUE VERDE

     COMPRENDO
AHORA ESTOY PONIENDO EL BLOQUE ROJO
ENCIMA DEL BLOQUE VERDE


        ......
        ....R.
        ....V.
        ....A.
        ....N.
```

Llegamos a una prueba muy difícil; sinceramente, yo pensé que el programa no sería capaz de realizarla. Es cierto que yo había escrito una rutina de manipulación blanco/objeto como objetivo general del programa, pero no estaba totalmente seguro de que éste fuera capaz de manipular una situación como la que está a punto de abordar ahora. Tiene que coger un bloque (el azul), quitando primero los bloques que están sobre él, y después ponerlo encima de uno de aquéllos (el verde):

```
? PON EL BLOQUE AZUL ENCIMA DEL VERDE

      COMPRENDO
DEBO MOVER EL BLOQUE ROJO
ESTOY MOVIENDOLO A LA COLUMNA  3
DEBO MOVER EL BLOQUE VERDE
ESTOY MOVIENDOLO A LA COLUMNA  6
AHORA ESTOY PONIENDO EL BLOQUE AZUL
ENCIMA DEL BLOQUE VERDE


        ......
        ......
        ......
        .....A
        ..R.NV
```

```
? DIME LO QUE VES

EMPEZANDO DESDE LA DERECHA
MIS SENSORES REGISTRAN EL BLOQUE AZUL
...Y MAS ABAJO DE EL...
VEO EL BLOQUE VERDE
MIS SENSORES REGISTRAN EL BLOQUE NARANJA
...UN ESPACIO LUEGO
ESTA EL BLOQUE ROJO
...UN ESPACIO LUEGO
FINALMENTE ...UN ESPACIO
```

Me sentí feliz cuando BLOQUELANDIA pasó esta prueba. No había pensado seguir sometiéndolo a pruebas tan difíciles, pero cuando lo intenté con una torre formada por los cuatro bloques, pidiéndole que sacara el bloque que estaba debajo del todo y lo pusiera encima, el programa estalló. Me llevó más de una semana el calcular cuál podría ser la solución. Primero lo intenté con un método ''klutz'', dándole una rutina diferente para las torres de cuatro bloques. Después me di cuenta de que sencillamente añadiéndole siete caracteres a la línea 1950 (TAREA = 3:) le permitía manipular la situación.

BLOQUELANDIA había trabajado mucho y bien, por lo que decidí que debería darle un respiro:

```
            ......
            ......
            ......
            .....A
            ..R.NV


? REVUELVE LOS BLOQUES
ESTA BIEN QUE LE DEN A UNO LA OPORTUNIDAD
    DE HACER LO QUE QUIERE


            ......
            ......
            ......
            ......
            VN.R.A
```

Pasamos ahora a estudiar los elementos más importantes del programa.

Módulos del programa

Al igual que otros programas de este libro, BLOQUELANDIA empieza con una llamada a una subrutina del final del programa que inicializa las variables utilizadas:

```
2580 REM INICIALIZACION
2590 CLS
2600 RANDOMIZE
2610 DIM A(5, 6)
2620 FOR X = 1 TO 5
2630 FOR Y = 1 TO 6
2640 A(X, Y) = 46
2650 NEXT Y
2660 NEXT X
2670 A(1, 2) = ASC("R"): REM BLOQUE ROJO
2680 A(1, 3) = ASC("N"): REM BLOQUE NARANJA
2690 A(1, 4) = ASC("A"): REM BLOQUE AZUL
2700 A(1, 5) = ASC("V"): REM BLOQUE VERDE
2710 RETURN
```

Como vemos (en la línea 2610), se utiliza una matriz de cinco por seis para contener el ''mundo''. Se rellena ésta, primero, con 46 líneas (620 a 2660), el código ASCII del punto que se utiliza para indicar un espacio vacío en el mundo del programa. La posición inicial de los bloques la dan las líneas comprendidas entre la 2670 y la 2700. El programa asigna la letra inicial del color ('R' para el rojo, etc.) al bloque de ese color. No hay nada verdaderamente complicado en esta primera subrutina.

Aunque la subrutina de inicialización es llamada sólo una vez durante toda la ejecución del programa, otra subrutina, NOMBRE DEL COLOR, será llamada cada vez que el ordenador quiera referirse a un bloque.

```
2510 REM NOMBRE DEL COLOR
2520 IF Q = ASC("R") THEN PRINT "ROJO"
2530 IF Q = ASC("N") THEN PRINT "NARANJA "
2540 IF Q = ASC("A") THEN PRINT "AZUL "
2550 IF Q = ASC("V") THEN PRINT "VERDE "
2560 RETURN
```

Esta subrutina cambia la letra inicial por el nombre completo del color en cuestión. Ambas subrutinas se encuentran al final del listado.

Volviendo al principio del programa, encontramos una corta sección de código que se encarga de imprimir una visión de conjunto de los bloques. Quizá esto podría haber sido una subrutina, pero como es necesaria cada vez que el programa pasa por el bucle principal, parece mejor tenerla aquí.

```
30 REM ** IMPRIMIR EL CUADRO **
40 CLS: PRINT: PRINT
50 FOR X = 5 TO 1 STEP - 1
60 PRINT TAB(8);
70 FOR Y = 1 TO 6
```

```
 80 PRINT CHR$(A(X, Y));
 90 NEXT Y
100 PRINT
110 NEXT X
120 PRINT: PRINT
```

La línea 50 muestra que la visión de conjunto se imprime de ‘‘arriba a abajo’’ apareciendo la hilera 5 antes que la 4 y así sucesivamente, hasta acabar en la hilera 1, en la parte inferior de la escena. Se hizo así para que al programa le resultara más fácil la manipulación de los bloques. Sabe que tiene que comprobar un número más alto para ver si hay un bloque encima del que está considerando en ese momento.

En realidad, no hubiera habido ninguna dificultad en hacerlo al revés (cuanto más bajo sea el número más arriba estará situado el bloque), pero me pareció que esto suponía una complicación innecesaria.

La siguiente sección del código acepta la entrada del usuario y partiendo de ésta determina qué subrutina ha de ser llamada.

```
130 INPUT A$
140 PRINT
150 IF A$ = "" THEN PRINT "FIN DEL PROGRAMA": END
    : REM TERMINA LA EJECUCION EN VEZ DE TENER
    QUE PRESIONAR LA TECLA DE 'RETURN'
160 IF LEFT$(A$, 10) = "DONDE ESTA" THEN GOSUB
    240
170 IF LEFT$(A$, 11) = "DIME LO QUE" THEN GOSUB
    1040
180 IF LEFT$(A$, 8) = "REVUELVE" THEN GOSUB 1260
190 IF LEFT$(A$, 6) = "PON EL" THEN GOSUB 1480
200 PRINT: PRINT: PRINT "      >>>> PULSE 'RETURN'
         PARA CONTINUAR": INPUT Z$
210 GOTO 40
```

Cuando uno se mete a construir un programa de inteligencia artificial enseguida cae en la cuenta de lo complejo que es el funcionamiento de la inteligencia. BLOQUELANDIA opera en un dominio muy restringido y reacciona sólo ante aquellas situaciones que se han tenido en cuenta específicamente (aunque, como vimos en la ejecución de muestra hace un momento, se las arregló para resolver una situación para la que yo pensaba que no lo había preparado). Pese a las limitaciones de dominio y de actuación, BLOQUELANDIA exige un montón de código, con secciones independientes que llevan a cabo cada investigación y siguen cada orden.

Observemos, por ejemplo, la rutina que determina la localización de un bloque en particular. En primer lugar, el programa verifica si el bloque por el que se le está preguntando forma parte de su propio universo. Lleva a cabo esta verificación extrayendo —con la ayuda de la línea 260— la inicial del bloque en cuestión, tras lo cual comprueba —línea 270 y 290— que es uno de los cuatro que él reconoce:

```
240 REM "DONDE ESTA EL BLOQUE..."
250 P = 0
260 B$ = MID$(A$, 15, 1)
270 IF B$ = "R" OR B$ = "N" OR B$ = "A" OR
    B$ = "V" THEN 350
280 B$ = MID$(A$, 22, 1)
290 IF B$ = "R" OR B$ = "N" OR B$ = "A" OR
    B$ = "V" THEN 350
```

Si, por ejemplo, le preguntáramos algo sobre un bloque rosa, usaría la rutina siguiente (entre las líneas 300 y 330) para escoger una respuesta al azar, antes de volver al programa principal:

```
300 IF RND(1) > .7 THEN 320
310 PRINT "LO SIENTO, NO TENGO INFORMACION DE ESO
         ": GOTO 330
320 PRINT "NO TENGO DATOS CON LOS QUE PODER RES
         PONDERTE"
330 RETURN
```

Si se salta esta rutina, empezará a buscar el bloque. No utilizamos bucles FOR/NEXT para esta búsqueda, ya que queremos que el programa sea capaz de dejar la búsqueda en cualquier momento. No estará bien construido un programa que salga arbitrariamente de los bucles FOR/NEXT (y muchos ordenadores se cuelgan bastante pronto si se le acumulan en la pila demasiadas direcciones NEXT). Esta parte del programa le da la primera parte de la información requerida por éste en relación con la localización del bloque:

```
350 M = ASC(B$)
360 PRINT TAB(8); "> DEJAME VER AHORA <"
370 X = 5
380 Y = 1
390 IF A(X, Y) = M THEN 430
400 IF Y < 6 THEN Y = Y + 1: GOTO 390
410 IF X > 1 THEN X = X - 1: GOTO 380
420 GOTO 300
430 IF X > 1 THEN 900: REM ESTA ENCIMA DE OTRO
440 IF Y > 1 THEN 540: REM NO ESTA EN LA IZQUIER
    DA
```

Las sentencias REM en el resto de esta sección explican lo que hace cada una:

```
460 REM ** EN LA IZQUIERDA **
470 PRINT "ESTA EN LA IZQUIERDA"
480 IF A(1, 2) = 46 THEN PRINT "NO HAY NADA INME
    DIATAMENTE A SU IZQUIERDA": GOTO 790
490 Q = A(1, 2)
500 PRINT
510 PRINT "A SU LADO, PUEDO VER EL BLOQUE "
520 GOSUB 2510
530 GOTO 790
540 IF Y < 6 THEN 650
550 REM ******************
560 REM ** EN LA DERECHA **
570 PRINT
580 PRINT "ESTA EN EL LADO DERECHO"
590 IF A(1, 5) = 46 THEN PRINT "NO HAY NADA INME
    DIATAMENTE A SU DERECHA": GOTO 790
600 PRINT "A SU IZQUIERDA PUEDO VER EL BLOQUE ";
610 Q = A(1, 5)
620 GOSUB 2510
630 GOTO 790
640 REM *****************
650 REM ** EN EL CENTRO **
660 PRINT
670 PRINT "ESTA EN LA CASILLA "; Y; "DE LA IZ
           QUIERDA"
680 IF A(X, Y - 1) = 46 THEN PRINT "NO HAY NADA
    INMEDIATAMENTE A SU IZQUIERDA": GOTO 730
690 Q = A(X, Y - 1)
700 PRINT "EL BLOQUE ";
710 GOSUB 2510
720 PRINT "ESTA INMEDIATAMENTE A SU IZQUIERDA"
730 IF A(X, Y + 1) = 46 THEN PRINT "NO HAY NADA
    INMEDIATAMENTE A SU DERECHA": GOTO 790
740 Q = A(X, Y + 1)
750 PRINT: PRINT "PUEDO VER EL BLOQUE ";
760 GOSUB 2510
770 PRINT " QUE LE ESTA TOCANDO POR LA DERECHA"
780 REM **************************
790 REM ** HAY ALGO POR ENCIMA? **
800 PRINT
810 P = X
820 IF X = 5 THEN 900
830 IF A(X + 1, Y) = 46 THEN PRINT: PRINT "NO HAY
    NADA POR ENCIMA DE EL": GOTO 330
840 PRINT: PRINT "ENCIMA ESTA EL BLOQUE ";
850 Q = A(X + 1, Y)
860 GOSUB 2510
870 X = X + 1
880 GOTO 820
890 REM **************************
900 REM ** ESTA ENCIMA DE OTRO? **
910 IF P <> 0 THEN X = P
920 PRINT
930 IF X = 1 THEN 330
940 PRINT "ESTA INMEDIATAMENTE ";
```

```
 950 PRINT "ENCIMA DEL BLOQUE ";
 960 Q = A(X - 1, Y)
 970 GOSUB 2510
 980 X = X - 1
 990 IF X < 2 THEN 330
1000 PRINT "QUE ESTA ";
1010 GOTO 950
1020 RETURN
```

Se llama a la siguiente subrutina si le preguntamos al programa DIME LO QUE VES. Esta es una rutina mucho más sencilla que la que localiza un bloque en particular.

```
1030 REM **********************
1040 REM ** DIME LO QUE VES **
1050 PRINT "EMPEZANDO DESDE LA DERECHA"
1060 Y = 6
1070 X = 5
1080 IF A(X, Y) <> 46 THEN 1140
1090 IF Y = 1 AND X = 1 THEN PRINT "FINALMENTE ";
1100 IF X = 1 AND A(X, Y) = 46 THEN PRINT "...UN
     ESPACIO ";: IF Y > 1 THEN PRINT "LUEGO"
1110 IF X > 1 THEN X = X - 1: GOTO 1080
1120 IF Y > 1 THEN Y = Y - 1: GOTO 1070
1130 RETURN
1140 L = INT(RND(1) * 2)
1150 IF L = 0 THEN PRINT "ESTA EL BLOQUE ";: GOTO
     1180
1160 IF L = 1 THEN PRINT "MIS SENSORES REGISTRAN
     EL BLOQUE ";: GOTO 1180
1170 PRINT "VEO EL BLOQUE ";
1180 Q = A(X, Y)
1190 GOSUB 2510
1200 IF X = 1 THEN 1120
1210 X = X - 1
1220 PRINT "...Y MAS ABAJO DE EL..."
1230 GOTO 1170
1240 RETURN
```

La rutina REVUELVE LOS BLOQUES es también relativamente sencilla:

```
1250 REM ***************************
1260 REM ** REVUELVE LOS BLOQUES **
1270 PRINT
1280 IF RND(1) > .5 THEN PRINT TAB(7); "YA ERA
     HORA, TAMBIEN": GOTO 1300
1290 PRINT "ESTA BIEN QUE LE DEN A UNO LA OPORTUNI
           DAD": PRINT TAB(4); "DE HACER LO QUE
           QUIERE"
```

```
1300 FOR X = 1 TO 5
1310 FOR Y = 1 TO 6
1320 A(X, Y) = 46
1330 NEXT Y
1340 NEXT X
1350 Y1 = INT(RND(1) * 6) + 1
1360 Y2 = INT(RND(1) * 6) + 1
1370 IF Y2 = Y1 THEN 1360
1380 Y3 = INT(RND(1) * 6) + 1
1390 IF Y3 = Y2 OR Y3 = Y1 THEN 1380
1400 Y4 = INT(RND(1) * 6) + 1
1410 IF Y4 = Y3 OR Y4 = Y2 OR Y4 = Y1 THEN 1400
1420 A(1, Y1) = 82
1430 A(1, Y2) = 78
1440 A(1, Y3) = 65
1450 A(1, Y4) = 86
1460 RETURN
```

Finalmente, llegamos a la rutina que produce los resultados más impresionantes, la rutina ‘‘PON EL objeto ENCIMA DEL blanco’’. Como en todas las rutinas importantes que hemos examinado, las sentencias REM explican lo que hace cada sección:

```
1480 REM "PON EL BLOQUE...ENCIMA DEL BLOQUE..."
1490 L = 0
1500 IF RND(1) > .5 THEN PRINT TAB(5); "COMPRENDO"
     : GOTO 1520
1510 PRINT TAB(8); "DE ACUERDO"
1520 IF MID$(A$, 8, 1) <> "B" THEN GOTO 1760: REM
     COMPRUEBA Y BIFURCA EN CASO DE HABERSE OMITI
     DO EL PRIMER 'BLOQUE'
1530 REM ** CASO: ''BLOQUE-COLOR...BLOQUE-COLOR''"
1540 B$ = MID$(A$, 15, 1)
1550 IF B$ = "R" THEN L = 38
1560 IF B$ = "A" THEN L = 38
1570 IF B$ = "V" THEN L = 39
1580 IF B$ = "N" THEN L = 41
1590 C$ = MID$(A$, L, 1)
1600 IF NOT (C$ = "R" OR C$ = "A" OR C$ = "V" OR
     C$ = "N") THEN 1840: REM COMPRUEBA Y BIFURCA
     CASO DE HABERSE  OMITIDO EL SEGUNDO BLOQUE
1610 B = ASC(B$)
1620 C = ASC(C$)
1630 BANDERA = C
1640 REM ** ENCUENTRA EL BLOQUE B$ **
1650 X = 5
1660 Y = 1
1670 IF A(X, Y) = B THEN 1900
1680 IF Y < 6 THEN Y = Y + 1: GOTO 1670
1690 IF X > 1 THEN X = X - 1: GOTO 1660
1700 PRINT "NO PUEDO ENCONTRAR EL BLOQUE ";
1710 Q = B
1720 GOSUB 2610
```

```
1730 FOR T = 1 TO 2000: NEXT T
1740 RETURN
1750 REM ** CASO: ''------COLOR...------COLOR'' **
1760 B$ = MID$(A$, 8, 1)
1770 IF B$ = "R" THEN L = 24
1780 IF B$ = "A" THEN L = 24
1790 IF B$ = "V" THEN L = 25
1800 IF B$ = "N" THEN L = 27
1810 IF NOT (L = 24 OR L = 25 OR L = 27) GOTO 1840
     : REM COMPRUEBA Y BIFURCA EN EL CASO DE HABER
     PUESTO EL SEGUNDO 'BLOQUE'
1820 GOTO 1590
1830 REM ** CASOS: ''BLOQUE-COLOR...------COLOR''
         Y ''------COLOR...BLOQUE-COLOR''  **
1840 IF B$ = "R" THEN L = 31
1850 IF B$ = "A" THEN L = 31
1860 IF B$ = "V" THEN L = 32
1870 IF B$ = "N" THEN L = 34
1880 C$ = MID$(A$, L, 1)
1890 GOTO 1610
1900 R = X: S = Y
1910 REM ** EL BLOQUE ESTA EN R, S **
1920 REM ** ESTA VACIA LA CASILLA DE DESTINO? **
1930 IF A(R + 1, S) = 46 THEN 2070: REM 'SI'
1940 IF A(R + 2, S) = 46 THEN TAREA = 1: GOTO 1960
1950 TAREA = 3: IF A(R + 3, S) = 46 THEN TAREA = 2
1960 FOR W = TAREA TO 1 STEP - 1
1970 PRINT "DEBO MOVER EL BLOQUE "
1980 Q = A(R + W, S)
1990 GOSUB 2510
2000 DE = INT(RND(1) * 6) + 1
2010 IF DE = S OR A(1, DE) = C OR A(2, DE) = C OR
     A(3, DE) = C THEN 2000
2020 PRINT "ESTOY MOVIENDOLO A LA COLUMNA "; DE
2030 L = 1
2040 IF A(L, DE) = 46 THEN A(L, DE) = A(R + W, S):
A(R + W, S) = 46: GOTO 2060
2050 L = L + 1: GOTO 2040
2060 NEXT W
2070 REM LA CASILLA DE DESTINO EN R, S ESTA AHORA
         LIBRE
2080 REM ** ESTA LIBRE LA CASILLA DE PARTIDA? **
2090 REM *** ENCUENTRA EL BLOQUE ***
2100 X = 5
2110 Y = 1
2120 IF A(X, Y) = C THEN 2210
2130 IF Y < 6 THEN Y = Y + 1: GOTO 2120
2140 IF X > 1 THEN X = X - 1: GOTO 2110
2150 PRINT "NO PUEDO ENCONTRAR EL BLOQUE ";
2160 Q = C
2170 GOSUB 2510
2180 FOR J = 1 TO 2000: NEXT J
2190 RETURN
2200 REM ** SE HA ENCONTRADO C ***
2210 T = X: U = Y: REM LOCALIZACION DE C
2220 IF A(T + 1, U) = 46 THEN 2390
2230 IF A(T + 2, U) = 46 THEN TAREA = 1: GOTO 2250
```

```
2240 IF A(T + 3, U) = 46 THEN TAREA = 2
2250 DE =INT(RND(1) * 6) + 1
2260 IF DE = U OR DE = S THEN 2250
2270 FOR W = TAREA TO 1 STEP - 1
2280 PRINT "AHORA MOVERE EL BLOQUE ";
2290 Q = A(T + W, U)
2300 GOSUB 2510
2310 PRINT
2320 PRINT "ESTOY MOVIENDOLO A LA COLUMNA "; DE
2330 L = 1
2340 IF A(L, DE) = 46 THEN A(L, DE) = A(T + W, U):
     A(T + W, U) = 46: GOTO 2360
2350 L =  L + 1: GOTO 2340
2360 NEXT W
2370 REM ** LA CASILLA DE PARTIDA AHORA ESTA LIBRE
        **
2380 REM ** REALIZA EL MOVIMIENTO **
2390 PRINT "AHORA ESTOY PONIENDO EL BLOQUE ";
2400 Q = A(R, S): Z = A(R, S)
2410 GOSUB 2510
2420 PRINT "ENCIMA DEL BLOQUE ";
2430 IF A(T, U) = 46 THEN A(T, U) = BANDERA
2440 Q = A(T, U)
2450 GOSUB 2510
2460 A(R, S) = 46
2470 A(T + 1, U) = Z
2480 FOR J = 1 TO 2000: NEXT J
2490 RETURN
```

Naturalmente, el programa tiene que atender a todas las situaciones que se le pide que manipule. Después del listado completo de éste, volveremos a "escuchar" un poco más de la conversación de Winograd con SHRDLU, de donde se pueden sacar ideas en el caso de que se quiera ampliar BLOQUELANDIA. Manteniendo una estructura similar a la presente, podemos añadir al programa un grado mayor de complejidad sin por ello perdernos en un laberinto de códigos.

La única información suplementaria que se necesita es el formato de la entrada exigido por el programa. Son tres las preguntas o peticiones que el usuario puede hacer al programa (y el programa espera que se las haga en mayúsculas, aunque esto se puede modificar dependiendo de los sistemas). Las preguntas son las siguientes:

DONDE ESTA EL BLOQUE (o CUBO, o como se le quiera llamar) seguido del color.

DIME LO QUE VES (o LO QUE PUEDES VER)

REVUELVE LOS BLOQUES

Podemos detener el programa en cualquier momento, sencillamente pulsando el RETURN (como se indica en la línea 150), cuando estemos preparados para hacer una pregunta o dar una orden.

Y aquí tenemos el listado completo de BLOQUELANDIA (para usuarios del ZX Spectrum, véase Apéndice):

```
 10 REM BLOQUELANDIA
 20 GOSUB 2580: REM INICIALIZACION
 30 REM ** IMPRIMIR EL CUADRO **
 40 CLS: PRINT: PRINT
 50 FOR X = 5 TO 1 STEP - 1
 60 PRINT TAB(8);
 70 FOR Y = 1 TO 6
 80 PRINT CHR$(A(X, Y));
 90 NEXT Y
100 PRINT
110 NEXT X
120 PRINT: PRINT
130 INPUT A$
140 PRINT
150 IF A$ = "" THEN PRINT "FIN DEL PROGRAMA": END
    : REM TERMINA LA EJECUCION EN VEZ DE TENER
    QUE PRESIONAR LA TECLA DE 'RETURN'
160 IF LEFT$(A$, 10) = "DONDE ESTA" THEN GOSUB
    240
170 IF LEFT$(A$, 11) = "DIME LO QUE" THEN GOSUB
    1040
180 IF LEFT$(A$, 8) = "REVUELVE" THEN GOSUB 1260
190 IF LEFT$(A$, 6) = "PON EL" THEN GOSUB 1480
200 PRINT: PRINT: PRINT "     >>>> PULSE 'RETURN'
         PARA CONTINUAR": INPUT Z$
210 GOTO 40
220 END
230 REM *************************
240 REM "DONDE ESTA EL BLOQUE..."
250 P = 0
260 B$ = MID$(A$, 15, 1)
270 IF B$ = "R" OR B$ = "N" OR B$ = "A" OR
    B$ = "V" THEN 350
280 B$ = MID$(A$, 22, 1)
290 IF B$ = "R" OR B$ = "N" OR B$ = "A" OR
    B$ = "V" THEN 350
300 IF RND(1) > .7 THEN 320
310 PRINT "LO SIENTO, NO TENGO INFORMACION DE ESO
           ": GOTO 330
320 PRINT "NO TENGO DATOS CON LOS QUE PODER RES
           PONDERTE"
330 RETURN
340 REM ***********************
350 M = ASC(B$)
360 PRINT TAB(8); "> DEJAME VER AHORA <"
370 X = 5
380 Y = 1
390 IF A(X, Y) = M THEN 430
400 IF Y < 6 THEN Y = Y + 1: GOTO 390
410 IF X > 1 THEN X = X - 1: GOTO 380
420 GOTO 300
430 IF X > 1 THEN 900: REM ESTA ENCIMA DE OTRO
440 IF Y > 1 THEN 540: REM NO ESTA EN LA IZQUIER
    DA
450 REM *********************
```

```
460 REM ** EN LA IZQUIERDA **
470 PRINT "ESTA EN LA IZQUIERDA"
480 IF A(1, 2) = 46 THEN PRINT "NO HAY NADA INME
    DIATAMENTE A SU IZQUIERDA": GOTO 790
490 Q = A(1, 2)
500 PRINT
510 PRINT "A SU LADO, PUEDO VER EL BLOQUE "
520 GOSUB 2510
530 GOTO 790
540 IF Y < 6 THEN 650
550 REM ******************
560 REM ** EN LA DERECHA **
570 PRINT
580 PRINT "ESTA EN EL LADO DERECHO"
590 IF A(1, 5) = 46 THEN PRINT "NO HAY NADA INME
    DIATAMENTE A SU DERECHA": GOTO 790
600 PRINT "A SU IZQUIERDA PUEDO VER EL BLOQUE ";
610 Q = A(1, 5)
620 GOSUB 2510
630 GOTO 790
640 REM ******************
650 REM ** EN EL CENTRO **
660 PRINT
670 PRINT "ESTA EN LA CASILLA "; Y; "DE LA IZ
           QUIERDA"
680 IF A(X, Y - 1) = 46 THEN PRINT "NO HAY NADA
    INMEDIATAMENTE A SU IZQUIERDA": GOTO 730
690 Q = A(X, Y - 1)
700 PRINT "EL BLOQUE ";
710 GOSUB 2510
720 PRINT "ESTA INMEDIATAMENTE A SU IZQUIERDA"
730 IF A(X, Y + 1) = 46 THEN PRINT "NO HAY NADA
    INMEDIATAMENTE A SU DERECHA": GOTO 790
740 Q = A(X, Y + 1)
750 PRINT: PRINT "PUEDO VER EL BLOQUE ";
760 GOSUB 2510
770 PRINT " QUE LE ESTA TOCANDO POR LA DERECHA"
780 REM *************************
790 REM ** HAY ALGO POR ENCIMA? **
800 PRINT
810 P = X
820 IF X = 5 THEN 900
830 IF A(X + 1, Y) = 46 THEN PRINT: PRINT "NO HAY
    NADA POR ENCIMA DE EL": GOTO 330
840 PRINT: PRINT "ENCIMA ESTA EL BLOQUE ";
850 Q = A(X + 1, Y)
860 GOSUB 2510
870 X = X + 1
880 GOTO 820
890 REM *************************
900 REM ** ESTA ENCIMA DE OTRO? **
910 IF P <> 0 THEN X = P
920 PRINT
930 IF X = 1 THEN 330
940 PRINT "ESTA INMEDIATAMENTE ";
950 PRINT "ENCIMA DEL BLOQUE ";
960 Q = A(X - 1, Y)
```

```
 970 GOSUB 2510
 980 X = X - 1
 990 IF X < 2 THEN 330
1000 PRINT "QUE ESTA ";
1010 GOTO 950
1020 RETURN
1030 REM ********************
1040 REM ** DIME LO QUE VES **
1050 PRINT "EMPEZANDO DESDE LA DERECHA"
1060 Y = 6
1070 X = 5
1080 IF A(X, Y) <> 46 THEN 1140
1090 IF Y = 1 AND X = 1 THEN PRINT "FINALMENTE ";
1100 IF X = 1 AND A(X, Y) = 46 THEN PRINT "...UN
     ESPACIO ";: IF Y > 1 THEN PRINT "LUEGO"
1110 IF X > 1 THEN X = X - 1: GOTO 1080
1120 IF Y > 1 THEN Y = Y - 1: GOTO 1070
1130 RETURN
1140 L = INT(RND(1) * 2)
1150 IF L = 0 THEN PRINT "ESTA EL BLOQUE ";: GOTO
     1180
1160 IF L = 1 THEN PRINT "MIS SENSORES REGISTRAN
     EL BLOQUE ";: GOTO 1180
1170 PRINT "VEO EL BLOQUE ";
1180 Q = A(X, Y)
1190 GOSUB 2510
1200 IF X = 1 THEN 1120
1210 X = X - 1
1220 PRINT "...Y MAS ABAJO DE EL..."
1230 GOTO 1170
1240 RETURN
1250 REM **************************
1260 REM ** REVUELVE LOS BLOQUES **
1270 PRINT
1280 IF RND(1) > .5 THEN PRINT TAB(7); "YA ERA
     HORA, TAMBIEN": GOTO 1300
1290 PRINT "ESTA BIEN QUE LE DEN A UNO LA OPORTUNI
            DAD": PRINT TAB(4); "DE HACER LO QUE
            QUIERE"
1300 FOR X = 1 TO 5
1310 FOR Y = 1 TO 6
1320 A(X, Y) = 46
1330 NEXT Y
1340 NEXT X
1350 Y1 = INT(RND(1) * 6) + 1
1360 Y2 = INT(RND(1) * 6) + 1
1370 IF Y2 = Y1 THEN 1360
1380 Y3 = INT(RND(1) * 6) + 1
1390 IF Y3 = Y2 OR Y3 = Y1 THEN 1380
1400 Y4 = INT(RND(1) * 6) + 1
1410 IF Y4 = Y3 OR Y4 = Y2 OR Y4 = Y1 THEN 1400
1420 A(1, Y1) = 82
1430 A(1, Y2) = 78
1440 A(1, Y3) = 65
1450 A(1, Y4) = 86
1460 RETURN
1470 REM ****************************************
```

```
1480 REM "PON EL BLOQUE...ENCIMA DEL BLOQUE..."
1490 L = 0
1500 IF RND(1) > .5 THEN PRINT TAB(5); "COMPRENDO"
     : GOTO 1520
1510 PRINT TAB(8); "DE ACUERDO"
1520 IF MID$(A$, 8, 1) <> "B" THEN GOTO 1760: REM
     COMPRUEBA Y BIFURCA EN CASO DE HABERSE OMITI
     DO EL PRIMER 'BLOQUE'
1530 REM ** CASO: ''BLOQUE-COLOR...BLOQUE-COLOR''"
1540 B$ = MID$(A$, 15, 1)
1550 IF B$ = "R" THEN L = 38
1560 IF B$ = "A" THEN L = 38
1570 IF B$ = "V" THEN L = 39
1580 IF B$ = "N" THEN L = 41
1590 C$ = MID$(A$, L, 1)
1600 IF NOT (C$ = "R" OR C$ = "A" OR C$ = "V" OR
     C$ = "N") THEN 1840: REM COMPRUEBA Y BIFURCA
     CASO DE HABERSE  OMITIDO EL SEGUNDO BLOQUE
1610 B = ASC(B$)
1620 C = ASC(C$)
1630 BANDERA = C
1640 REM ** ENCUENTRA EL BLOQUE B$ **
1650 X = 5
1660 Y = 1
1670 IF A(X, Y) = B THEN 1900
1680 IF Y < 6 THEN Y = Y + 1: GOTO 1670
1690 IF X > 1 THEN X = X - 1: GOTO 1660
1700 PRINT "NO PUEDO ENCONTRAR EL BLOQUE ";
1710 Q = B
1720 GOSUB 2610
1730 FOR T = 1 TO 2000: NEXT T
1740 RETURN
1750 REM ** CASO: ''------COLOR...------COLOR'' **
1760 B$ = MID$(A$, 8, 1)
1770 IF B$ = "R" THEN L = 24
1780 IF B$ = "A" THEN L = 24
1790 IF B$ = "V" THEN L = 25
1800 IF B$ = "N" THEN L = 27
1810 IF NOT (L = 24 OR L = 25 OR L = 27) GOTO 1840
     : REM COMPRUEBA Y BIFURCA EN EL CASO DE HABER
     PUESTO EL SEGUNDO 'BLOQUE'
1820 GOTO 1590
1830 REM ** CASOS: ''BLOQUE-COLOR...------COLOR''
         Y ''------COLOR...BLOQUE-COLOR''   **
1840 IF B$ = "R" THEN L = 31
1850 IF B$ = "A" THEN L = 31
1860 IF B$ = "V" THEN L = 32
1870 IF B$ = "N" THEN L = 34
1880 C$ = MID$(A$, L, 1)
1890 GOTO 1610
1900 R = X: S = Y
1910 REM ** EL BLOQUE ESTA EN R, S **
1920 REM ** ESTA VACIA LA CASILLA DE DESTINO? **
1930 IF A(R + 1, S) = 46 THEN 2070: REM 'SI'
1940 IF A(R + 2, S) = 46 THEN TAREA = 1: GOTO 1960
1950 TAREA = 3: IF A(R + 3, S) = 46 THEN TAREA = 2
1960 FOR W = TAREA TO 1 STEP - 1
```

```
1970 PRINT "DEBO MOVER EL BLOQUE "
1980 Q = A(R + W, S)
1990 GOSUB 2510
2000 DE = INT(RND(1) * 6) + 1
2010 IF DE = S OR A(1, DE) = C OR A(2, DE) = C OR
     A(3, DE) = C THEN 2000
2020 PRINT "ESTOY MOVIENDOLO A LA COLUMNA "; DE
2030 L = 1
2040 IF A(L, DE) = 46 THEN A(L, DE) = A(R + W, S):
A(R + W, S) = 46: GOTO 2060
2050 L = L + 1: GOTO 2040
2060 NEXT W
2070 REM LA CASILLA DE DESTINO EN R, S ESTA AHORA
        LIBRE
2080 REM ** ESTA LIBRE LA CASILLA DE PARTIDA? **
2090 REM *** ENCUENTRA EL BLOQUE ***
2100 X = 5
2110 Y = 1
2120 IF A(X, Y) = C THEN 2210
2130 IF Y < 6 THEN Y = Y + 1: GOTO 2120
2140 IF X > 1 THEN X = X - 1: GOTO 2110
2150 PRINT "NO PUEDO ENCONTRAR EL BLOQUE ";
2160 Q = C
2170 GOSUB 2510
2180 FOR J = 1 TO 2000: NEXT J
2190 RETURN
2200 REM ** SE HA ENCONTRADO C ***
2210 T = X: U = Y: REM LOCALIZACION DE C
2220 IF A(T + 1, U) = 46 THEN 2390
2230 IF A(T + 2, U) = 46 THEN TAREA = 1: GOTO 2250
2240 IF A(T + 3, U) = 46 THEN TAREA = 2
2250 DE =INT(RND(1) * 6) + 1
2260 IF DE = U OR DE = S THEN 2250
2270 FOR W = TAREA TO 1 STEP - 1
2280 PRINT "AHORA MOVERE EL BLOQUE ";
2290 Q = A(T + W, U)
2300 GOSUB 2510
2310 PRINT
2320 PRINT "ESTOY MOVIENDOLO A LA COLUMNA "; DE
2330 L = 1
2340 IF A(L, DE) = 46 THEN A(L, DE) = A(T + W, U):
     A(T + W, U) = 46: GOTO 2360
2350 L =  L + 1: GOTO 2340
2360 NEXT W
2370 REM ** LA CASILLA DE PARTIDA AHORA ESTA LIBRE
        **
2380 REM ** REALIZA EL MOVIMIENTO **
2390 PRINT "AHORA ESTOY PONIENDO EL BLOQUE ";
2400 Q = A(R, S): Z = A(R, S)
2410 GOSUB 2510
2420 PRINT "ENCIMA DEL BLOQUE ";
2430 IF A(T, U) = 46 THEN A(T, U) = BANDERA
2440 Q = A(T, U)
2450 GOSUB 2510
2460 A(R, S) = 46
2470 A(T + 1, U) = Z
2480 FOR J = 1 TO 2000: NEXT J
```

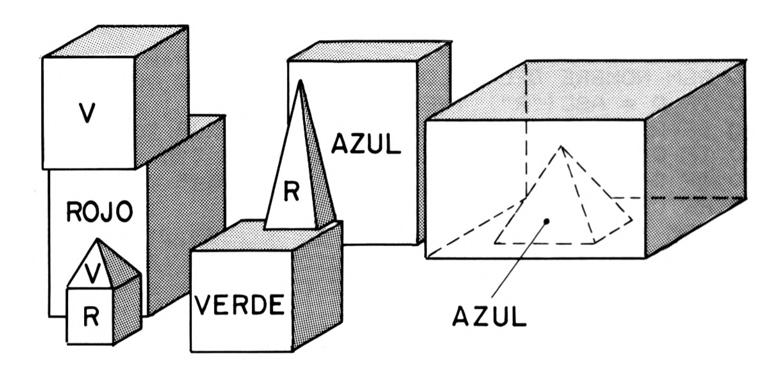
```
2490 RETURN
2500 REM ****************
2510 REM NOMBRE DEL COLOR
2520 IF Q = ASC("R") THEN PRINT "ROJO"
2530 IF Q = ASC("N") THEN PRINT "NARANJA "
2540 IF Q = ASC("A") THEN PRINT "AZUL "
2550 IF Q = ASC("V") THEN PRINT "VERDE "
2560 RETURN
2570 REM **************
2580 REM INICIALIZACION
2590 CLS
2600 RANDOMIZE
2610 DIM A(5, 6)
2620 FOR X = 1 TO 5
2630 FOR Y = 1 TO 6
2640 A(X, Y) = 46
2650 NEXT Y
2660 NEXT X
2670 A(1, 2) = ASC("R"): REM BLOQUE ROJO
2680 A(1, 3) = ASC("N"): REM BLOQUE NARANJA
2690 A(1, 4) = ASC("A"): REM BLOQUE AZUL
2700 A(1, 5) = ASC("V"): REM BLOQUE VERDE
2710 RETURN
```

Para mejorar el programa

Aunque el rendimiendo de BLOQUELANDIA en su forma presente ya es bastante encomiable, todavía se puede hacer mucho por él. Como hemos visto, consta de una sencilla estructura modular en la que una sola subrutina se encarga de cada tipo de orden. Así pues, para aumentar sus habilidades, podemos ampliar la sección de reconocimiento del vocabulario del inicio del programa y utilizarla después para poner en acción nuevas subrutinas.

Podríamos empezar, por ejemplo, dándole al programa la capacidad de mover los bloques hacia la derecha o hacia la izquierda, en vez de sólo encima o debajo de otros bloques como lo hace ahora. Una vez que hayamos conseguido esto (asegurándonos asimismo de que el programa cuenta con un medio de rechazar las peticiones "imposibles", como la de intentar que ponga un bloque a la derecha de otro que ya está totalmente a la derecha del marco), podríamos tratar de añadir un quinto bloque o, quizá, otra figura diferente —o más de una—. Finalmente, cada uno puede crear enteramente, empezando por el principio, su propio programa, haciendo que los bloques y otros elementos se muevan en tres dimensiones, de modo que "delante" y "detrás" fueran órdenes válidas.

Con el fin de que el lector pueda sacar nuevas ideas sobre lo que puede hacerse, volveremos a considerar ahora el programa decisivo en este campo, SHRDLU, escrito por Terry Winograd. Lo que vimos antes, en este mismo capítulo, era el inicio de la conversación que viene a continuación. En primer lugar, éste es el croquis de la situación sobre la que hablan programador y programa:



Dejemos ahora que continúe la conversación:

```
Coge el bloque rojo grande
 > DE ACUERDO
Agarra la piramide
 > NO COMPRENDO LO QUE ME DICES DE LA
     PIRAMIDE
Busca el bloque que es mas alto que el
   que tienes, e introducelo en la caja.
 > POR "LO" , SUPONGO QUE TE REFIERES
      AL BLOQUE QUE ES MAS ALTO QUE
      EL QUE TENGO
 > DE ACUERDO
Que es lo que hay en la caja?
 > LA PIRAMIDE AZUL Y EL BLOQUE AZUL
Quien sostiene a la piramide?
 > LA CAJA
Cuantos bloques no estan en la caja?
 > CUATRO
Hay al menos uno de ellos mas estrecho
   que el que te dije que cogieras?
 > SI, EL CUBO ROJO
Le sostiene alguien?
 > SI, LA MESA
Puede la mesa recoger bloques?
 > NO
Puede un bloque sostener a una piramide?
 > SI
Puede una piramide sostener a otra piramide?
 > NO LO SE
Levanta dos piramides
 > NO PUEDO
La piramide azul es mia
 > COMPRENDO
```

Y esto no es nada comparado con lo que vamos a ''oír'' ahora. Se incluye aquí una manipulación lógica que incluso los seres humanos encontrarían difícil de seguir. Sigue así la conversación:

```
Tengo todos los bloques que no son rojos,
  pero no tengo nada con que sostener a una
  piramide.
 > COMPRENDO
Tengo yo la caja?
 > NO
Tengo algo en la caja?
 > SI, DOS COSAS: EL BLOQUE AZUL Y LA PIRAMI
     DE AZUL
Que cubo esta encima de la mesa?
 > EL VERDE GRANDE QUE SOSTIENE A LA PIRAMIDE
     ROJA
Hay algun bloque grande detras de la
  piramide?
 > SI, HAY TRES: UN CUBO ROJO GRANDE, UNO
     VERDE GRANDE Y EL AZUL
Pon el pequeno dentro del cubo verde que
  sostiene a la piramide.
 > DE ACUERDO
Pon la piramide mas pequena encima de el
 > DE ACUERDO
Cuantas cosas hay encima de los cubos verdes?
 > NO ESTOY MUY SEGURO DE LO QUE QUIERES
     DECIR CON "ENCIMA DE" EN LA FRASE
     "ENCIMA DE LOS CUBOS VERDES"
 > QUIERES DECIR:
  1 - DIRECTAMENTE SOBRE LA SUPERFICIE
  2 - ENCIMA DE CUALQUIER PARTE?
Has tocado alguna piramide antes de poner el
  cubo verde sobre el cubo pequeno?
 > SI, EL VERDE
Cuando lo cogiste?
 > CUANDO ESTABA GUARDANDO EL CUBO ROJO, EL
     ROJO GRANDE Y EL VERDE GRANDE
Por que?
 > PARA DESHACERME DE EL
Por que querias hacer eso?
 > PARA DEPEJAR EL CUBO ROJO
Por que querias despejar ese cubo?
 > PARA PONER SOBRE EL EL CUBO VERDE GRANDE
Por que hiciste eso?
 > PORQUE ME LO PEDISTE
```

¿Qué más podríamos añadir? Ahora que ya has visto lo que SHRDLU y BLOQUELANDIA son capaces de hacer, ponte manos a la obra y haz tú mismo, lector, algo parecido.



8
El médico en casa

Ningún estudio sobre el ''procesado de una lengua natural'' sería un estudio completo si no incluyera un comentario sobre ELIZA, el programa más famoso de este tipo (y, en un momento determinado, el programa más famoso de todos).

ELIZA fue creado entre 1964 y 1966 por Joseph Weizenbaum, a la sazón profesor de Informática del Institute of Technology en Massachusetts. La idea era que el programa adoptara el papel de un psiquiatra de la escuela de Carl Rogers (Weizenbaum siempre insistió en que su intención era parodiar a los profesionales de esta escuela). El método de trabajo de un psiquiatra rogeriano consiste en negarse a dirigir la conversación con el paciente, prefiriendo, en vez, reflejar en éste sus propias observaciones, de tal modo que la conversación siempre depende del paciente.

La reacción ante el programa —una vez que se hizo público— primero asombró y después pasó a preocupar seriamente a Weizenbaum. A resultas de esto, Weizenbaum dejó el Institute por dos años para meditar sobre las profundas implicaciones de su experiencia con ELIZA, y durante este tiempo escribió una de sus obras más importantes (importantísima también para el campo de la informática en general); se trata de *Computer Power and Human Reason* (San Francisco, W. H. Freeman and Co., 1976). Yo no estoy de acuerdo con todo lo que se dice en esta obra, pero esto no disminuye la importancia del libro en sí mismo. Independientemente de mis propias opiniones al respecto, es un libro cuya lectura recomiendo fervorosamente a todos aquellos que estén de verdad interesados en el estudio de la inteligencia artificial y en las relaciones y dependencias que se desarrollan entre el hombre y la máquina.

El programa de Weizenbaum se utiliza tecleando por nuestro lado una ''conversación'' a la que el programa responde utilizando su almacén de respuestas. Aquí tenemos un ejemplo de una conversación típica, sacado de la ejecución de ELIZA que se da en este libro:

```
>? YO ESTOY CON MIEDO POR HABLAR EN PUBLICO

CREE QUE ES NORMAL ESTAR
CON MIEDO POR HABLAR EN PUBLICO

---------------------------------------------

>? PIENSO QUE UN MONTON DE GENTE LO TIENE

PIENSA REALMENTE ESO?

---------------------------------------------

>? SI LO PIENSO

PARECE MUY SEGURO DE ELLO

---------------------------------------------

>? HABLARLE A UN ORDENADOR ES BASTANTE ABURRIDO

LO PIENSA SOBRE MI EN PARTICULAR?
```

Weizenbaum escogió el nombre de ELIZA porque pensó que, como en el caso de la Eliza de *Pygmalion*, éste iría aprendiendo a hablar cada vez mejor. El programa ELIZA original se escribió en dos partes. La primera analizaba la entrada del usuario y la segunda era un "guión". Se idearon diferentes guiones para los diferentes temas que podrían tratarse y DOCTOR era uno de estos guiones. (Otros guiones podían mantener conversaciones sobre buques antiguos, el cambio de la moneda o lo que fuera.)

El DOCTOR (un psiquiatra rogeriano) fue el primer guión que probó Weizenbaum. El programa enseguida se hizo famoso en el Institute porque era un medio muy efectivo de demostrar el poder de un ordenador (recordemos que esto sucedía hace mucho, mucho tiempo, en una galaxia lejana, cuando la gente no podía comprarse un ordenador en la tienda del ramo más próxima a su domicilio).

A su debido tiempo, Weizenbaum informó de su trabajo sobre ELIZA a la prensa especializada ("ELIZA. A Computer Program For the Study of Natural Language Communication Between Man and Machine", *Communications of the Association for Computing Machinery,* vol. 9, núm. 1, enero 1965, pp. 36-45) y no tardó en haber un gran número de versiones —basadas en su descripción— repartidas por otras instituciones de Estados Unidos.

Weizenbaum nos dice que fundamentalmente fueron tres los acontecimientos que le chocaron al irse propagando el uso de ELIZA. En primer lugar, se quedó horrorizado (yo no llego a comprender por qué se alarmó tanto como nos dice) al descubrir que la gente inmediatamente se sentía involucrada en el programa.

Cuenta también Weizenbaum que incluso su secretaria, que había trabajado con él en el desarrollo del programa durante varios meses, y que, por tanto, debería estar en situación de darse cuenta de que era sólo un programa, empezó a tener una relación emocional con éste. Siempre cuenta Weizenbaum cómo en una ocasión esta misma secretaria empezó a usar el programa y cuando no llevaría introducidas más de unas cuantas frases se quedó callada y, muy avergonzada, le pidió que saliera un momento de la habitación mientras ella continuaba la "conversación".

En otra ocasión Weizenbaum sugirió la idea de poner una impresora para tener una

transcripción de las conversaciones que la gente tenía con el programa. La idea fue acogida con espanto por parte de todo el mundo, ya que significaba que Weizenbaum podría fisgoner en conversaciones totalmente privadas.

Una de las cosas que más le preocupaban era la fuerza con la que la gente se identificaba con el programa, dándole una personalidad, y compartía con él sus pensamiento, más íntimos. Dijo que no había calculado el "poder de engaño" que un programa, bastante simple, por otro lado, podía crear en la gente normal.

La conexión rusa

Pamela McCorduck, en su espléndido libro *Machines Who Think* (San Francisco, W. H. Freeman and Co., 1979), confirma el efecto que puede tener el programa. Nos cuenta que la primera vez que vio ELIZA ejecutado y charlando fue en el Stanford Computation Center con motivo de la visita de un científico soviético, muy conocido internacionalmente en el campo de la informática, a quien le estaban enseñando las instalaciones del centro.

Este se sentó ante el ordenador, que estaba conectado a una versión del programa escrita por uno de los colegas de Weizenbaum, Kenneth Colby (a quien volveremos a encontrar más adelante), y empezó a teclear. McCorduck nos cuenta la vergüenza que llegó a pasar cuando, lanzado ante la frase CUENTAME ALGO DE TU FAMILIA, el científico empezó a comentar en profundidad ciertos problemas personales, olvidándose de todos los que le rodeaban.

Weizenbaum descubrió que algunos accesos al programa, mediante las terminales a tiempo compartido que había repartidas por la universidad, a menudo llegaban a durar una hora o más y muy frecuentemente se hacían a altas horas de la madrugada. Recibía asimismo llamadas telefonías de gente que quería acceder al programa durante un rato pensando que así solucionarían sus problemas.

Colby, a quien acabamos de mencionar, había conocido a Weizenbaum algún tiempo antes, en Stanford. Era profesor de Psiquiatría en la Universidad de California y estaba interesado en la inteligencia artificial. Pensaba que los hallazgos que se hicieran en ésta podrían llevar a nuevas visiones sobre el pensamiento humano (y, además, Colby esperaba sacar nuevas ideas aplicables a sus estudios sobre el comportamiento neurótico). Antes de que apareciera el primer trabajo de Weizenbaum sobre ELIZA, Colby publicó una breve notificación sobre este tema en el *Journal of Nervous and Mental Diseases*.

Esto provocó la separación de ambos; en primer lugar, porque Weizenbaum no estaba de acuerdo en aboluto con las tesis de Colby, para quien el programa podría tener aplicaciones terapéuticas, pero además porque, al parecer, Colby no citó a Weizenbaum por su trabajo en el ELIZA.

Colby y dos colegas más opinaban que una versión mejorada de DOCTOR podría tener un genuino uso terapéutico. Pensaban que se podría poner a disposición de los hospitales que tuvieran problemas de escasez de personal, de modo que los pacientes pudieran pedir acceder al programa (vía un sistema de terminales compartidas) cada vez que lo consideraran necesario. A Weizenbaum todas estas ideas le aterraban. Dice que consideraba vital que hubiera, como punto de partida para que una persona pudiera ayudar a otra a enfrentarse con sus problemas, un reconocimiento de éstos por parte de otro humano.

Weizenbaum no daba crédito a sus ojos al ver que incluso un psiquiatra era de la opinión de que el proceso curativo podría sustituirse por la mera técnica mecánica. Semejante idea nunca había pasado por su cabeza. Es más, aun en el caso de que pudiera hacerse, no debería hacerse. Hay ciertas áreas a las que nunca se debería permitir que accedieran las máquinas, opinaba Weizenbaum, aun cuando éstas estuvieran capacitadas para hacerlo.

Colby no escarmentó con la respuesta de Weizenbaum. Al parecer, no tenía ningún inconveniente en considerar la posibilidad de que la técnica pura podría ser eficaz en su campo. Y seguía defendiendo su opinión diciendo que sólo los legos confundían la psicoterapia con el matrimonio. Según él, lo que importaba era la relación profesional, de trabajo, entre un psicoterapeuta y un paciente.

Más exactamente, Colby atacaba la afirmación de Weizenbaum de que había áreas en las que nunca se deberían utilizar las máquinas. ¿Por qué no?, preguntaba Colby. ¿Sólo porque el señor Weizenbaum dice que no? ¿Es que cree que ayudar a la gente con el ordenador es peor que dejarla sufrir? ¿Acaso no debe explorar el terapeuta todos los medios a su alcance, aunque sólo sea para comprobar si alguno de ellos resulta ser genuinamente efectivo?

Las opiniones de Colby las secunda en mayor o menor grado Carl Sagan, quien últimamente piensa en desarrollar la idea de poner a disposición del público un programa parecido al ELIZA —a unos cuantos dólares la sesión— en habitáculos especialmente construidos para ello, algo parecido a las cabinas telefónicas (*Broca's Brain,* Londres, Coronet Books, 1980, p. 300).

Independientemente del "valor real" del bueno de DOCTOR (como quiera que lo juzguemos), no hay ninguna duda de que el programa marcó un hito significativo en la historia de la producción de programas que parecen mostrar cierta inteligencia.

Y aquí es donde Weizenbaum volvió a recibir un *shock*. Recordemos que primero se había quedado asombrado ante la inequívoca antropomorfización que sufrió el programa. Luego pasó a alarmarse seriamente cuando se empezó a sugerir que ELIZA podría ocupar el lugar o, al menos, ayudar a los terapeutas humanos. El tercer *shock* le vino al observar que mucha gente se llegó a creer de verdad que el programa era en cierto modo una demostración palpable de que se estaba a un paso de que las máquinas llegaran a comprender el lenguaje humano. El había rechazado totalmente esta idea. En realidad, en el primer artículo que publicó sobre el programa, Weizenbaum hizo gran hincapié en la idea de que era imposible encontrar una solución general para este problema.

Decía antes que no estaba de acuerdo con todo lo que Weizenbaum afirma en su libro *Computer Power and Human Reason*. Uno de los puntos para mí más discutibles es la afirmación de que "hay ciertas cosas que nunca deberían ser hechas por las máquinas". John McCarthy (1976, "An Unreasonable Book", en *Three Reviews of J. Weizenbaum's Computer Power and Human Reason,* Memo AIM-291, Stanford AI Laboratory, noviembre) es de la opinión de que si hay funciones que no se deben enseñar ni dejar que lleve a cabo un ordenador, se trata de funciones que no se deberían realizar en absoluto, ya se trate de una persona o de un ordenador.

Son más de uno los que comparten esta misma opinión. En su libro *Artificial Reality* (1983, Addison-Wesley Publishing Co., Reading, MA, p. 168), Myron W. Krueger dice

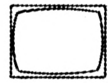
que aun cuando fuera verdad que Weizenbaum se sintió horrorizado con la idea de utilizar su programa —o un desarrollo de éste— en ciertos tipos de terapia, tal horror carecía de fundamento y justificación.

No obstante, al margen de lo que yo (y otros) pueda pensar de las tesis de Weizenbaum y del valor de su libro (ya dije antes que creo que es un libro que deberían leer todos los interesados en el tema, aunque sólo fuera para aportar su granito de arena a este debate), no cabe duda de que tanto ELIZA como DOCTOR han demostrado ser unos compañeros extremadamente divertidos. El lector lo comprobará enseguida por sí mismo.

Y no sólo se divertirá, sino que además estará en situación de juzgar si de verdad el programa sugiere que la inteligencia reside en la máquina que lo está ejecutando. Adrian Berry (en su libro *The Super-Intelligent Machine,* Londres, Jonathan Cape Ltd., 1983, página 63) llega a la conclusión de que ELIZA (y PARRY, un programa ideado para imitar a un paciente paranoico) no demuestra nada en cuanto a la posibilidad de que pueda llegar a darse una verdadera inteligencia.

El mismo lector observará que sus opiniones oscilan según usa el programa. Cuando DOCTOR haga alguna observación particularmente inspirada y apropiada al momento, sentirá que ésta es una prueba clara de inteligencia, al menos a cierto nivel. En otras ocasiones, descubrirá que DOCTOR es ideal sencillamente para pasar un buen rato, ya que lo que dice, a veces, no tiene ni pies ni cabeza. Nadie nos haría unas preguntas tan absurdas (¿ACUDIO A MI PORQUE YO ESTABA MUERTO?, le preguntó una vez una ejecución del programa a Berry, según nos cuenta éste en su libro) o sacaría tan inocentemente unas conclusiones no por erróneas menos soberbias, como el trocito de conversación que viene a continuación, que forma parte de la ejecución de muestra del programa que incluimos en este capítulo (DOCTOR habla en mayúsculas):

```
Siento que usted esta eludiendo mi pregunta
  SIENTE A MENUDO QUE YO ESTOY ELUDIENDO SU PREGUNTA
Lo siento cuando hablo con un ordenador
  LAS COSAS SUELEN OCURRIR CASI SIEMPRE EN EL MOMENTO OPORTUNO
```



¿Qué otra cosa podemos hacer ante una afirmación tal sino alzar un brindis por Joseph W., progenitor de ELIZA y de todos sus retoños, y esperar que nos siga ofreciendo más cosas de este tipo? Creo que ahora ha llegado el momento de que dediquemos un poco de atención al programa en sí mismo.

El programa

Una vez que ya sabemos cómo se escribió ELIZA y cuál fue la reacción que provocó su versión de DOCTOR, echemos un vistazo al funcionamiento de nuestra propia versión del programa. Lo que viene a continuación es una ejecución de éste. La entrada de la persona va precedida por el signo >? y una línea de puntos separa cada una de las afirmaciones del humano y la respuesta del ordenador del siguiente par de afirmación/respuesta:

```
EL DOCTOR ESTA EN LA CONSULTA...POR FAVOR ESPERE

BIENVENIDO A OTRA SESION
CON EL DOCTOR...

HOLA...

----------------------------------------------

>? YO ESTOY CON MIEDO POR HABLAR EN PUBLICO

CREE QUE ES NORMAL ESTAR
CON MIEDO POR HABLAR EN PUBLICO

----------------------------------------------

>? PIENSO QUE UN MONTON DE GENTE LO TIENE

PIENSA REALMENTE ESO?

----------------------------------------------

>? SI LO PIENSO

PARECE MUY SEGURO DE ELLO

----------------------------------------------

>? HABLARLE A UN ORDENADOR ES BASTANTE ABURRIDO

LO PIENSA SOBRE MI EN PARTICULAR?

----------------------------------------------

>? VIENE DE MI LUCHA CONTINUA DIA Y NOCHE

AL PRINCIPIO ME HABLO DE SU PROFESOR LOCO
CUENTEME ALGO MAS

----------------------------------------------

>? EMPEZO CUANDO TENIA DIEZ ANOS

ESO ES MUY INTERESANTE

----------------------------------------------

>? ESTOY CON MIEDO DE IR A LA ESCUELA AHORA

ACUDIO A MI PORQUE ESTA
CON MIEDO DE IR A LA ESCUELA AHORA

----------------------------------------------

>? CREO QUE ES VERDAD

PERO NO ESTA SEGURO QUE ES VERDAD

----------------------------------------------

>? TAL VEZ HAY OTRAS RAZONES

ESO ES MUY INTERESANTE

----------------------------------------------
```

```
>? TAMBIEN TENGO PROBLEMAS CON MI HERMANO

NO SEA TAN ESCUETO, POR FAVOR ACLARELO UN POCO MAS

----------------------------------------------

>? CREO QUE EL ME ODIA

DUDA QUE EL LE ODIA

----------------------------------------------

>? SIENTO QUE USTED ESTA ELUDIENDO MI PREGUNTA

SIENTE A MENUDO QUE YO ESTOY ELUDIENDO SU PREGUNTA

----------------------------------------------

>? LO SIENTO CUANDO HABLO CON UN ORDENADOR

LAS COSAS SUELEN OCURRIR CASI SIEMPRE EN EL MOMENTO OPORTUNO

----------------------------------------------

>? ESA ES UNA RESPUESTA ESTUPIDA DE UN ORDENADOR

QUE ASPECTOS DE LOS ORDENADORES LE INTERESAN MAS?

----------------------------------------------

>? LOS QUE ME AYUDEN EN MIS PROBLEMAS

NO SEA TAN ESCUETO, POR FAVOR ACLARELO UN POCO MAS

----------------------------------------------

>? ME GUSTARIA SACUDIRLE A MI HERMANO

QUE SUPONDRIA EN SU VIDA SI USTED CONSIGUIERA
SACUDIRLE A SU HERMANO

----------------------------------------------

>? ME HARIA SENTIRME MUY BIEN

VAMOS A EXAMINAR AFIRMACION MAS DETENIDAMENTE

----------------------------------------------

>? ME SIENTO INFERIOR CUANDO EL ME HABLA

DISFRUTA SINTIENDOSE INFERIOR CUANDO EL LE HABLA

----------------------------------------------

>? PUES CLARO QUE NO DISFRUTO

LE SUGIERE QUE EXISTE ALGUNA CONEXION
ENTRE ESO Y SU HERMANO?

----------------------------------------------

>? LA RELACION ESTA CLARA PARA CUALQUIERA

NO SEA TAN ESCUETO, POR FAVOR ACLARELO UN POCO MAS
```

```
----------------------------------------------

>? QUIERO ALGO DE AYUDA CON MIS APRENSIONES

QUE SUPONDRIA EN SU VIDA SI CONSIGUIERA
ALGO DE AYUDA CON SUS APRENSIONES

----------------------------------------------

>? CREO QUE ME SENTIRIA MUCHO MEJOR

DUDA QUE LE SENTIRIA MUCHO MEJOR

----------------------------------------------

>? SIEMPRE HE QUERIDO SER LIBRE

NO LE PARECE QUE 'SIEMPRE' ES UN POCO EXAGERADO?

----------------------------------------------

>? CREO QUE SI

POR QUE NO RETROCEDEMOS UN POCO
Y HABLAMOS POCO MAS DE SU APRENSIONES?

----------------------------------------------
```

Esta ejecución de muestra es bastante impresionante y estoy seguro de que en cuanto el lector haya usado el programa unas cuantas veces descubrirá el tipo de frases y palabras a las que mejor responde el programa. Asimismo, como ya señalaba antes, es una demostración excelente para aquellos que no están muy acostumbrados al funcionamiento de los ordenadores. (Pero se ha de tener cuidado no vaya a ser que entre en juego ese "poder de engaño" del que hablaba Weizenbaum y nos encontremos con que uno de nuestros amigos empieza a confiar a nuestra máquina sus secretos más íntimos.)

Funcionamiento

El programa es, en esencia, sencillo. El ordenador acepta una entrada en cadena alfanumérica introducida por el usuario y después la examina buscando las "frases clave" (tales como CREO o ¿POR QUE NO PUEDO?) para las que tiene un *stock* de respuestas. Si no consigue encontrar ninguna de estas frases, buscará entonces "palabras clave" (como ORDENADOR) y, del mismo modo, escoge una de las tres respuestas preprogramadas para esa clave.

Si no encuentra ninguna clave (ni frase, ni palabra), mirará si tiene almacenada alguna frase del usuario de una entrada anterior (como ACTITUD o PROFESOR CHIFLADO), y si la tiene, crea una pregunta partiendo de esta información. Si así tampoco consigue encontrar una respuesta, escogerá entre una serie de respuestas aleatorias (como NO ESTOY SEGURO DE ENTENDER ESO POR COMPLETO o algo similar) que se programaron con la idea de mantener viva la conversación.

```
AL PRINCIPIO ME HABLO DE SU ACTITUD
CUENTEME ALGO MAS

----------------------------------------------

>? ES UNA PREOCUPACION PARA MI

QUE GRADO DE CERTEZA DEPOSITARIA EN ELLO SIENDO
UNA PREOCUPACION PARA USTED

----------------------------------------------

>? MUCHA PORQUE LA TIENE

QUE OTRAS RAZONES RECUERDA?
```

El banco de respuestas

La parte más importante del programa, y la más divertida a la hora de modificarla o ampliarla, es el inmenso banco de sentencias DATA que aparece al final de éste. (Comprobará el lector que las últimas sentencias DATA son una serie de asteriscos. Se ha hecho así para terminar la lectura y permitir que el usuario pueda añadir más DATA, independientemente de los que ya haya. Las sentencias DIM para que las matrices de cadenas alfanuméricas contengan estos DATA son más grandes de lo que debieran ser, con el fin de que el usuario pueda añadir más palabras, si así lo desea.)

Veremos que el primer conjunto de sentencias DATA cubre la conjugación de los verbos y la declinación de los pronombres:

```
1410 REM ** DATOS **
1420 REM ** CONJUGACIONES **
1430 DATA SOY, ES, ERA, ERA, YO, USTED, MI, SU,
     MIS, SUS
1440 DATA "USTED TIENE", "YO TENGO", SU, MI, ES,
     SOY, ME, LE
1450 DATA "USTED ESTA", "YO ESTOY", SUYO, MIO,
     USTED, MI, MI, USTED, *, *
```

El ordenador los usa exactamente igual a como lo hace en el caso de TRADUCTOR; es decir, maneja el español intercambiando cada palabra por su pareja. Esto significa lo siguiente: si por ejemplo le decimos ESTOY RECIBIENDO TU MENSAJE, el ordenador sencillamente intercambiará pronombres y verbos, respondiendo ESTAS RECIBIENDO MI MENSAJE. De hecho, esta es la forma que adoptaba el programa DOCTOR original y se puede decir que incluso este limitado tipo de intercambio puede ser muy significativo.

Tras esto vienen las sentencias DATA principales, que se cuidan del intercambio de la mayoría de las frases. Las hay de dos tipos. El primer tipo usa ya sea una palabra o

una frase corta (utilizada al inicio de la entrada del usuario) y después escoge una respuesta completa del banco de respuestas, sin tomar ninguna palabra directamente de la entrada del usuario (es decir, no responde utilizando ninguna de ellas). En estos ejemplos, las primeras sentencias DATA de cada una de las cuatro es la "clave" sacada de la entrada del usuario y las tres siguientes son aquéllas de entre las cuales el ordenador escoge su respuesta:

```
1830 DATA "COMO"
1840 DATA "COMO ARREGLARIA ESO?"
1850 DATA "LO MEJOR SERIA QUE SE RESPONDIERA A ESO
     USTED MISMO"
1860 DATA "QUE ES LO QUE REALMENTE ESTA PREGUNTAN
     DO?"


2030 DATA "PORQUE"
2040 DATA "ES ESA LA VERDADERA RAZON?"
2050 DATA "QUE OTRAS RAZONES RECUERDA?"
2060 DATA "QUE MAS PUEDE EXPLICAR ESO?"
```

Un mayor interés tienen las "frases clave" que se usan como inicio de la respuesta del ordenador, siendo la respuesta el resto de la entrada original del usuario (después de hacer los cambios necesarios en la conjugación):

```
1470 DATA "NECESITO"
1480 DATA "POR QUE NECESITA*"
1490 DATA "NO SERIA REALMENTE PROVECHOSO SI USTED*
     "
1500 DATA "ESTA SEGURO DE QUE NECESITA*"


1670 DATA "YO ESTOY"
1680 DATA "ACUDIO A MI PORQUE ESTA*"
1690 DATA "CUANTO TIEMPO HA ESTADO*"
1700 DATA "CREE QUE ES NORMAL ESTAR*"


2510 DATA "ESTA"
2520 DATA "PIENSA QUE ESTAR*"
2530 DATA "EN QUE CIRCUNSTANCIAS ESTARIA*"
2540 DATA "BIEN PODRIA SER QUE*"
```

Hemos de observar que cada una de las frases que forman parte de la respuesta terminan con "*", lo que utiliza el ordenador a modo de bandera o señalizador para indicar que esa parte de la entrada original ha de ser modificada para completar la sentencia.

Veamos cómo funciona en la práctica. Supongamos que la entrada del usuario fuera la siguiente:

```
QUIERO MOSTRARLE LA VERDAD
```

El DOCTOR examina los contenidos de la matriz de cadenas alfanuméricas C$ y encuentra el elemento que contiene QUIERO. Los elementos equivalentes de las matrices D$, E$ y F$ contienen los inicios de las respuestas adecuadas, como vemos aquí:

```
1750 DATA "QUIERO"
1760 DATA "QUE SIGNIFICARIA PARA USTED SI*"
1770 DATA "POR QUE QUIERE*"
1780 DATA "QUE SUPONDRIA EN SU VIDA SI USTED CONSI
     GUIERA*"
```

El DOCTOR genera un número aleatorio entre uno y tres e imprime el elemento D$ si es uno, el E$ si es dos y el F$ si es tres (habiendo comprobado previamente que termina con un asterisco, y si es así, lo anota y después lo quita antes de imprimir). Supongamos que el ordenador ha escogido la respuesta D$. Su respuesta sería entonces:

```
POR QUE QUIERE
```

Luego recorre el resto de la entrada del usuario (lo que sigue a QUIERO) tratándolo del mismo modo (usando el mismo código, de hecho) que lo hace el programa TRADUCTOR cuando cambia las palabras inglesas en palabras españolas; es decir, cambia cosas como USTED por YO y ES por SOY (de modo que YO SOY se convierte en USTED ES). Va imprimiendo cada palabra a medida que la procesa, dejando la palabra de la entrada del usuario igual, si no hay nada que necesite ser cambiado en la sección de conjugación y declinación.

La frase original...

```
QUIERO SALUDAR A SU PADRE
```

... se transforma en...

```
POR QUE QUIERE SALUDAR A MI PADRE
```

Esto es "todo" lo que hace el programa, pero el lector comprobará enseguida por sí mismo que el efecto que crea es notable.

Si no consigue encontrar una frase clave que pueda combinar con una parte de la entrada del usuario, el DOCTOR buscará una palabra clave situada en cualquier posi-

ción de la entrada de éste (en vez de hacerlo sólo al inicio, que es donde busca las frases). Entre las palabras clave se incluyen ORDENADOR y AMIGOS, que producen resultados como estos:

```
POR QUE TOCA AHORA EL TEMA DE LOS AMIGOS?

POR FAVOR HABLEME MAS SOBRE SUS AMISTADES...
```

... y...

```
QUE ASPECTOS DE LOS ORDENADORES LE INTERESAN MAS?

LO PIENSA SOBRE MI EN PARTICULAR?
```

La estructura del programa

El programa empieza, como el resto de los que componen este libro, con una llamada a una rutina de inicialización (la que empieza en la línea 1140). Tras esto, el programa imprime una línea en blanco y después una línea de puntos que separa cada par de preguntas y respuestas, seguida de otra línea en blanco.

```
10 REM DOCTOR
20 GOSUB 1140: REM INICIALIZACION
30 PRINT: PRINT "-------------------------------
        ---------------": PRINT
40 PRINT ">"; : INPUT X$
50 IF X$ = "" THEN PRINT "FIN DEL PROGRAMA"
60 PRINT
70 IF X$ = Z$ THEN PRINT "POR FAVOR...NO SE REPI
   TA": GOTO 30
80 Z$ = X$
90 IF LEFT$(X$,5) = "ADIOS" THEN PRINT "ADIOS,
   ESPERO VERLE DE NUEVO": END
```

La línea 40 imprime el signo recordatorio ''>'' y después acepta la entrada. Si la entrada es una cadena alfanumérica vacía (es decir, el usuario ha pulsado RETURN, en vez de introducir una frase), el programa termina.

La línea 60 imprime una línea en blanco y la línea 70 compara esta entrada (X$) con la anterior (Z$) y, si ve que son iguales, dice POR FAVOR... NO SE REPITA, tras lo cual vuelve a la línea 30 para recibir una nueva entrada. La línea 80 fija la nueva entrada igual a Z$, para comprobarla posteriormente. Si las primeras cinco letras de la entrada componen la palabra ADIOS, el ordenador responde con ADIOS, ESPERO QUE VOLVAMOS A VERNOS.

Habiendo sobrevivido a esta serie de obstáculos, empieza a trabajar en serio:

```
100 REM ** BUSQUEDA DE FRASES CLAVE AL PRINCIPIO
        DE LA INTRODUCCION **
110 L = 0
120 L = L + 1
130 LN = LEN(C$(L))
140 IF LEFT$(X$, LN) = C$(L) THEN 360: REM SE HA
    ENCONTRADO LA FRASE CLAVE
150 IF L < K THEN 120
```

El programa examina, utilizando para ello los elementos de la matriz C$, las primeras partes de la entrada, buscando una pareja. Si encuentra una, la acción se mueve a la línea 360, en donde se lleva a cabo la tarea de emparejar la frase de la entrada con el resto de la entrada del usuario:

```
360 REM SE HA ENCONTRADO UNA FRASE CLAVE AL PRIN
        CIPIO DE LA INTRODUCCION
370 T = INT(RND(1) * 3) + 1
380 IF T = 1 THEN G$ = D$(L)
390 IF T = 2 THEN G$ = E$(L)
400 IF T = 3 THEN G$ = F$(L)
410 REM ** SE VERIFICA SI LA FRASE ACABA EN UN AS
        TERISCO. DE SER ASI HAY QUE ANADIRLA ALGO
        MAS **
420 BANDERA = 0
430 IF RIGHT$(G$, 1) = "*" THEN BANDERA = 1: G$ =
    LEFT$(G$, (LEN(G$) - 1))
440 PRINT G$; " ";
450 IF BANDERA = 0 THEN 30: REM NO HAY QUE ANADIR
    NADA
460 REM ** AHORA SE UTILIZA LA FRASE INTRODUCIDA
        **
470 X$ = " " + MID$(X$, LN + 2, LEN(X$)) + " "
480 REM ****************************
490 REM ** CAMBIOS DE CONJUGACION **
500 REM ** TAMBIEN SE BUSCA 'MI' PARA LA CLAVE
        'MIBANDERA' (K$) **
510 LN = LEN(X$)
520 M = 0
530 M = M + 1
540 IF M = LN THEN 30
550 IF MID$(X$, M, 1) = " " THEN 570
560 GOTO 530
570 X = M + 1
580 Y = 0
590 Y = Y + 1
600 IF MID$(X$, (X + Y), 1) = " " THEN Q$ = MID$
    (X$, X, Y): GOTO 630
```

```
610 IF X + Y > 250 THEN 530
620 GOTO 590
630 MN = 0
640 MN = MN + 1
650 IF Q$ = "MI" AND K$ = "" THEN K$ = MID$(X$, X
    + 3, LEN(X$) - 4): K$ = LEFT$(K$, LEN(K$) - 1
    )
660 IF Q$ = A$(MN) THEN PRINT B$(MN); " ";: GOTO
    530
670 IF MN < KK THEN 640
680 PRINT Q$; " ";
690 GOTO 530
```

La sección comprendida entre las líneas 370 y 400 escoge uno de los tres inicios de respuesta entre las posibilidades D$(n), E$(n) y F$(n). La línea 420 pone a cero un señalizador (llamado BANDERA) y después utiliza la línea 430 para verificar que la frase escogida acaba con un asterisco (con esto recordamos que se trata sólo de una respuesta parcial que necesita más material de la entrada del usuario).

Si descubre que tiene un asterisco al final, la bandera se pone a uno y la parte final de la línea 430 lo quita.

La línea 440 imprime la frase escogida. Si BANDERA sigue siendo igual a cero (línea 450), el programa vuelve a la línea 30 para recibir la siguiente entrada. Si no, el DOCTOR ha de examinar el resto de la entrada del usuario, haciendo los cambios necesarios en la conjugación (y utilizando, como ya he dicho antes, el mismo código que el empleado en TRADUCTOR).

Asimismo, como lo indica la sentencia REM en la línea 500, el programa busca la palabra MI para disparar K$, el ''mibandera''. Si encuestra la palabra MI en la entrada (como en la frase MI PROFESOR CHIFLADO) y todavía no se le ha asignado nada al ''mibandera'' (K$), tomará el resto de la entrada del usuario a partir de la palabra MI y la asignará a K$, de modo que —en este caso, por ejemplo— se fijaría a PROFESOR CHIFLADO. Así, si más adelante DOCTOR no consiguiera encontrar una clave en la entrada del usuario, podría usar K$ con otras frases (como AL PRINCIPIO ME HABLO DE SU PROFESOR CHIFLADO. CUENTEME ALGO MAS) para mantener la conversación. El efecto que esta trampita produce en los usuarios del programa es extraordinario.

Si no ha encontrado ninguna frase clave, el ordenador busca una palabra clave, utilizando esta sección del código:

```
160 REM ** EL PROGRAMA LLEGA AQUI EN EL CASO DE
          QUE NO SE HAYA ENCONTRADO UNA FRASE CLAVE
          AL PRINCIPIO DE X$ **
170 REM ** AHORA SE BUSCAN PALABRAS CLAVE DENTRO
          DE TODA LA FRASE INTRODUCIDA **
180 X$ = " " + X$ + " "
190 M = LEN(X$)
200 L = 0
210 L = L + 1
```

```
220 IF L = M - 1 THEN 800: REM NO SE HA ENCONTRA
    DO UNA PALABRA CLAVE
230 IF MID$(X$, L, 1) = " " THEN 250
240 GOTO 210
250 X = L + 1
260 Y = 0
270 Y = Y + 1
280 IF MID$(X$, (X + Y), 1) = " " THEN Q$ = MID$(
    X$, X, Y): GOTO 300
290 GOTO 270
300 N = 0
310 N = N + 1
320 IF Q$ = C$(N) THEN 710: REM SE HA ENCONTRADO
    UNA PALABRA CLAVE
330 IF N < K THEN 310
340 GOTO 210
```

La utilización del "mibandera" (señalizador)

Si su búsqueda resulta infructuosa, el ordenador tendrá que recurrir de nuevo al "mibandera" (si hay uno) o a una frase aleatoria (si la bandera es una cadena sin asignar):

```
800 REM RESPUESTAS ALEATORIAS/NO CLAVES
810 IF K$ <> "" THEN 1010: REM 'MIBANDERA' NO
    ESTA VACIA, ASI QUE SE VA ALLI
820 T = INT(RND(1) * 8) + 1
830 ON T GOSUB 850, 870, 890, 910, 930, 950, 970,
    990
840 GOTO 30
850 PRINT "QUE LE SUGIERE A USTED?"
860 RETURN
870 PRINT "NO ESTOY SEGURO DE ENTENDER ESO POR
           COMPLETO"
880 RETURN
890 PRINT "NO SEA TAN ESCUETO, POR FAVOR ACLARELO
           UN POCO MAS"
900 RETURN
910 PRINT "ESO ES MUY INTERESANTE"
920 RETURN
930 PRINT "BIEN, BIEN...POR FAVOR CONTINUE..."
940 RETURN
950 PRINT "POR QUE?"
960 RETURN
970 PRINT "Y ENTONCES?"
980 RETURN
990 PRINT "YA VEO...POR FAVOR SIGA HABLANDO MAS
           SOBRE ELLO"
1000 RETURN
1010 REM ** UTILIZACION DE 'MIBANDERA' **
1020 T = INT(RND(1) * 8) + 1
```

```
1030 IF T = 1 THEN PRINT "HABLEME MAS SOBRE SU ";
     K$
1040 IF T = 2 THEN PRINT "AL PRINCIPIO ME HABLO DE
     SU "; K$: PRINT "CUENTEME ALGO MAS"
1050 IF T = 3 THEN PRINT "TIENE ALGO QUE VER":
     PRINT "CON SU"; K$; "?"LLIST 1050
1060 IF T = 4 THEN PRINT "EXISTE ALGUNA CONEXION
     CON": PRINT "SU "; K$; "?"
1070 IF T = 5 THEN PRINT "POR QUE NO RETROCEDEMOS
     UN POCO": PRINT "Y HABLAMOS MAS DE SU "; K$;
     "?"
1080 IF T = 6 THEN PRINT "LE SUGIERE QUE EXISTE AL
     GUNA CONEXION": PRINT "ENTRE ESO Y SU "; K$;
     "?"
1090 IF T = 7 THEN PRINT "PREFERIRIA HABLAR SOBRE
     SU "; K$; "?"
1100 IF T = 8 THEN PRINT "CREO QUE EL PREOCUPARSE
     SOBRE SU "; K$: PRINT "NO LE CONDUCE A NADA"
1110 IF RND(1) > .7 THEN K$ = ""
1120 GOTO 30
```

La línea 810 comprueba a qué se le ha asignado la cadena variable K$. Si descubre que K$ no está vacía, la acción pasa a la sección comprendida entre las líneas 1020 y 1110 y se imprime una de las ocho respuestas que puede usar el ‘‘mibandera’’ (tales como HABLEME MAS SOBRE o POR QUE NO RETROCEDEMOS UN POCO Y HABLAMOS UN POCO MAS DE SU...).

Al final de esta sección (línea 1110), K$ se vuelve a fijar a la cadena vacía en un 30 por 100 de las veces, de modo que pueda volver a fijarse a otra cosa diferente, si se vuelve a encontrar otro MI en una entrada posterior.

Si K$ no tiene nada asignado, el DOCTOR escoge entonces una de las ocho respuestas aleatorias (comprendidas en las líneas 850 a 1000). Estas están ideadas con el fin de que la conversación no se interrumpa.

Ahora que ya sabemos cómo funciona, ha llegado el momento de abrir nuestra consulta para empezar a ejercer de psiquiatras (para usuarios del ZX Spectrum, véase Apéndices).

```
  10 REM DOCTOR
  20 GOSUB 1140: REM INICIALIZACION
  30 PRINT: PRINT "--------------------------------
          ---------------": PRINT
  40 PRINT ">"; : INPUT X$
  50 IF X$ = "" THEN PRINT "FIN DEL PROGRAMA"
  60 PRINT
  70 IF X$ = Z$ THEN PRINT "POR FAVOR...NO SE REPI
     TA": GOTO 30
  80 Z$ = X$
  90 IF LEFT$(X$,5) = "ADIOS" THEN PRINT "ADIOS,
     ESPERO VERLE DE NUEVO": END
 100 REM ** BUSQUEDA DE FRASES CLAVE AL PRINCIPIO
          DE LA INTRODUCCION **
```

```
110 L = 0
120 L = L + 1
130 LN = LEN(C$(L))
140 IF LEFT$(X$, LN) = C$(L) THEN 360: REM SE HA
    ENCONTRADO LA FRASE CLAVE
150 IF L < K THEN 120
160 REM ** EL PROGRAMA LLEGA AQUI EN EL CASO DE
        QUE NO SE HAYA ENCONTRADO UNA FRASE CLAVE
        AL PRINCIPIO DE X$ **
170 REM ** AHORA SE BUSCAN PALABRAS CLAVE DENTRO
        DE TODA LA FRASE INTRODUCIDA **
180 X$ = " " + X$ + " "
190 M = LEN(X$)
200 L = 0
210 L = L + 1
220 IF L = M - 1 THEN 800: REM NO SE HA ENCONTRA
    DO UNA PALABRA CLAVE
230 IF MID$(X$, L, 1) = " " THEN 250
240 GOTO 210
250 X = L + 1
260 Y = 0
270 Y = Y + 1
280 IF MID$(X$, (X + Y), 1) = " " THEN Q$ = MID$(
    X$, X, Y): GOTO 300
290 GOTO 270
300 N = 0
310 N = N + 1
320 IF Q$ = C$(N) THEN 710: REM SE HA ENCONTRADO
    UNA PALABRA CLAVE
330 IF N < K THEN 310
340 GOTO 210
350 REM ****************************************
        ***********************
360 REM SE HA ENCONTRADO UNA FRASE CLAVE AL PRIN
        CIPIO DE LA INTRODUCCION
370 T = INT(RND(1) * 3) + 1
380 IF T = 1 THEN G$ = D$(L)
390 IF T = 2 THEN G$ = E$(L)
400 IF T = 3 THEN G$ = F$(L)
410 REM ** SE VERIFICA SI LA FRASE ACABA EN UN AS
        TERISCO. DE SER ASI HAY QUE ANADIRLA ALGO
        MAS **
420 BANDERA = 0
430 IF RIGHT$(G$, 1) = "*" THEN BANDERA = 1: G$ =
    LEFT$(G$, (LEN(G$) - 1))
440 PRINT G$; " ";
450 IF BANDERA = 0 THEN 30: REM NO HAY QUE ANADIR
    NADA
460 REM ** AHORA SE UTILIZA LA FRASE INTRODUCIDA
        **
470 X$ = " " + MID$(X$, LN + 2, LEN(X$)) + " "
480 REM ****************************
490 REM ** CAMBIOS DE CONJUGACION **
500 REM ** TAMBIEN SE BUSCA 'MI' PARA LA CLAVE
        'MIBANDERA' (K$) **
510 LN = LEN(X$)
520 M = 0
```

```
530 M = M + 1
540 IF M = LN THEN 30
550 IF MID$(X$, M, 1) = " " THEN 570
560 GOTO 530
570 X = M + 1
580 Y = 0
590 Y = Y + 1
600 IF MID$(X$, (X + Y), 1) = " " THEN Q$ = MID$
    (X$, X, Y): GOTO 630
610 IF X + Y > 250 THEN 530
620 GOTO 590
630 MN = 0
640 MN = MN + 1
650 IF Q$ = "MI" AND K$ = "" THEN K$ = MID$(X$, X
    + 3, LEN(X$) - 4): K$ = LEFT$(K$, LEN(K$) - 1
    )
660 IF Q$ = A$(MN) THEN PRINT B$(MN); " ";: GOTO
    530
670 IF MN < KK THEN 640
680 PRINT Q$; " ";
690 GOTO 530
700 REM ***********************
710 REM ENCUENTRA PALABRAS CLAVE
720 T = INT(RND(1) * 3) + 2
730 Q$ = ""
740 IF T = 1 THEN Q$ = D$(N)
750 IF T = 2 THEN Q$ = E$(N)
760 IF T = 3 THEN Q$ = F$(N)
770 IF RIGHT$(Q$, 1) <> "*" THEN PRINT Q$: GOTO
    30
780 REM ** SE PASA A LA SIGUIENTE SECCION SI LA
        PALABRA CLAVE SE CONSIDERA NO VALIDA **
790 REM ******************************
800 REM RESPUESTAS ALEATORIAS/NO CLAVES
810 IF K$ <> "" THEN 1010: REM 'MIBANDERA' NO
    ESTA VACIA, ASI QUE SE VA ALLI
820 T = INT(RND(1) * 8) + 1
830 ON T GOSUB 850, 870, 890, 910, 930, 950, 970,
    990
840 GOTO 30
850 PRINT "QUE LE SUGIERE A USTED?"
860 RETURN
870 PRINT "NO ESTOY SEGURO DE ENTENDER ESO POR
           COMPLETO"
880 RETURN
890 PRINT "NO SEA TAN ESCUETO, POR FAVOR ACLARELO
           UN POCO MAS"
900 RETURN
910 PRINT "ESO ES MUY INTERESANTE"
920 RETURN
930 PRINT "BIEN, BIEN...POR FAVOR CONTINUE..."
940 RETURN
950 PRINT "POR QUE?"
960 RETURN
970 PRINT "Y ENTONCES?"
980 RETURN
```

```
 990 PRINT "YA VEO...POR FAVOR SIGA HABLANDO MAS
              SOBRE ELLO"
1000 RETURN
1010 REM ** UTILIZACION DE 'MIBANDERA' **
1020 T = INT(RND(1) * 8) + 1
1030 IF T = 1 THEN PRINT "HABLEME MAS SOBRE SU ";
     K$
1040 IF T = 2 THEN PRINT "AL PRINCIPIO ME HABLO DE
     SU "; K$: PRINT "CUENTEME ALGO MAS"
1050 IF T = 3 THEN PRINT "TIENE ALGO QUE VER":
     PRINT "CON SU"; K$; "?"LLIST 1050
1060 IF T = 4 THEN PRINT "EXISTE ALGUNA CONEXION
     CON": PRINT "SU "; K$; "?"
1070 IF T = 5 THEN PRINT "POR QUE NO RETROCEDEMOS
     UN POCO": PRINT "Y HABLAMOS MAS DE SU "; K$;
     "?"
1080 IF T = 6 THEN PRINT "LE SUGIERE QUE EXISTE AL
     GUNA CONEXION": PRINT "ENTRE ESO Y SU "; K$;
     "?"
1090 IF T = 7 THEN PRINT "PREFERIRIA HABLAR SOBRE
     SU "; K$; "?"
1100 IF T = 8 THEN PRINT "CREO QUE EL PREOCUPARSE
     SOBRE SU "; K$: PRINT "NO LE CONDUCE A NADA"
1110 IF RND(1) > .7 THEN K$ = ""
1120 GOTO 30
1130 REM **************
1140 REM INICIALIZACION
1150 RANDOMIZE
1160 CLS
1170 DEFINT A - Z
1180 DIM A$(16), B$(16): REM CONJUGACIONES
1190 DIM C$(45), D$(45), E$(45), F$(45): REM PALA
     BRAS CLAVE Y RESPUESTAS
1200 Z$ = "": REM PARA FINALIZAR LAS REPETICIONES
1210 K$ = "": REM 'MIBANDERA'
1220 PRINT: PRINT
1230 PRINT "EL DOCTOR ESTA EN LA CONSULTA...POR FA
              VOR ESPERE"
1240 KK = 0
1250 KK = KK + 1
1260 READ A$(KK), B$(KK)
1270 IF B$(KK) = "*" THEN 1290
1280 GOTO 1250
1290 K = 0
1300 K = K + 1
1310 READ C$(K), D$(K), E$(K), F$(K)
1320 IF F$(K) = "*" THEN 1340
1330 GOTO 1300
1340 CLS
1350 PRINT "BIENVENIDO A OTRA SESION"
1360 PRINT "CON EL DOCTOR..."
1370 PRINT
1380 PRINT "HOLA..."
1390 RETURN
1400 REM ***********
1410 REM ** DATOS **
1420 REM ** CONJUGACIONES **
```

```
1430 DATA SOY, ES, ERA, ERA, YO, USTED, MI, SU,
     MIS, SUS
1440 DATA "USTED TIENE", "YO TENGO", SU, MI, ES,
     SOY, ME, LE
1450 DATA "USTED ESTA", "YO ESTOY", SUYO, MIO,
     USTED, MI, MI, USTED, *, *
1460 REM ** PALABRAS CLAVE/FRASES DE RESPUESTA **
1470 DATA "NECESITO"
1480 DATA "POR QUE NECESITA*"
1490 DATA "NO SERIA REALMENTE PROVECHOSO SI USTED*
     "
1500 DATA "ESTA SEGURO DE QUE NECESITA*"
1510 DATA "POR QUE NO*"
1520 DATA "PIENSA REALMENTE QUE YO NO*"
1530 DATA "QUIZAS FINALMENTE YO*"
1540 DATA "REALMENTE QUIERE QUE YO*"
1550 DATA "POR QUE NO PUEDO"
1560 DATA "CREE QUE USTED SERIA CAPAZ DE*"
1570 DATA "POR QUE NO PUEDE*"
1580 DATA "QUIZAS NO LO HAYA INTENTADO LO SUFICIEN
     TE"
1590 DATA "ESTA USTED"
1600 DATA "POR QUE ESTA INTERESADO EN SABER SI YO
     SOY O NO*"
1605 DATA *,*
1610 DATA "LO PREFERIRIA SI NO FUERA*"
1620 DATA "QUIZAS ALGUNAS VECES SUENA QUE SOY*"
1630 DATA "NO PUEDO"
1640 DATA "COMO SABE QUE NO PUEDE*"
1650 DATA "LO HA INTENTADO?"
1660 DATA "QUIZAS, AHORA, USTED PUEDA"
1670 DATA "YO ESTOY"
1680 DATA "ACUDIO A MI PORQUE ESTA*"
1690 DATA "CUANTO TIEMPO HA ESTADO*"
1700 DATA "CREE QUE ES NORMAL ESTAR*"
1710 DATA "SOY"
1720 DATA "DISFRUTA SIENDO*"
1730 DATA "POR QUE ME DICE QUE ES*"
1740 DATA "POR QUE ES USTED*"
1750 DATA "QUIERO"
1760 DATA "QUE SIGNIFICARIA PARA USTED SI*"
1770 DATA "POR QUE QUIERE*"
1780 DATA "QUE SUPONDRIA EN SU VIDA SI USTED CONSI
     GUIERA*"
1790 DATA "QUE"
1800 DATA "POR QUE PREGUNTA ESO?"
1810 DATA "QUE SE RESPONDERIA PARA AYUDARSE?"
1820 DATA "QUE PIENSA USTED?"
1830 DATA "COMO"
1840 DATA "COMO ARREGLARIA ESO?"
1850 DATA "LO MEJOR SERIA QUE SE RESPONDIERA A ESO
     USTED MISMO"
1860 DATA "QUE ES LO QUE REALMENTE ESTA PREGUNTAN
     DO?"
1870 DATA "QUIEN"
1880 DATA "SE HACE A MENUDO ESAS PREGUNTAS?"
1890 DATA "QUE RESPUESTA LE DEJARIA TRANQUILO?"
```

```
1900 DATA "QUIEN CREE USTED QUE*"
1910 DATA "DONDE"
1920 DATA "ESA ES UNA PREGUNTA BASTANTE INGENUA"
1930 DATA "NECESITA REALMENTE SABER DONDE?"
1940 DATA "QUE SUPONDRIA PARA USTED SI YO LE DIJE
     RA DONDE*"
1950 DATA "CUANDO"
1960 DATA "COMO PODRIA SABER CUANDO*"
1970 DATA "NO ESTAMOS AQUI PARA DISCUTIR SOBRE EL
     TIEMPO"
1980 DATA "LAS COSAS SUELEN OCURRIR CASI SIEMPRE
     EN EL MOMENTO OPORTUNO"LLIST 1980
1990 DATA "POR QUE"
2000 DATA "POR QUE NO ME DICE LA VERDADERA RAZON
     POR LA QUE*"
2010 DATA "QUE ES LO QUE HA DICHO QUE ME AYUDARIA
     A DECIRLE POR QUE*"
2020 DATA "NECESITA REALMENTE SABER POR QUE*"
2030 DATA "PORQUE"
2040 DATA "ES ESA LA VERDADERA RAZON?"
2050 DATA "QUE OTRAS RAZONES RECUERDA?"
2060 DATA "QUE MAS PUEDE EXPLICAR ESO?"
2070 DATA "LO SIENTO"
2080 DATA "EN QUE OTRAS CIRCUNSTANCIAS SE DISCULPA
     ?"
2090 DATA "HAY MUCHAS OCASIONES EN LAS QUE NO ES
     NECESARIO DISCULPARSE"
2100 DATA "QUE SENSACION EXPERIMENTA CUANDO TIENE
     QUE DISCULPARSE?"
2110 DATA "HOLA"
2120 DATA "HOLA...ME ALEGRO DE VERLE"
2130 DATA "HOLA...ME ALEGRO DE QUE SE HAYA DEJADO
     CAER HOY POR AQUI"
2140 DATA "COMO ESTA USTED?...TENIA GANAS DE QUE
     CHARLARAMOS DE NUEVO"
2150 DATA "OIGA!"
2160 DATA "ME ALEGRO DE VERLE HOY POR AQUI"
2170 DATA "ME ALEGRO DE QUE HAYA VENIDO. TENEMOS
     MONTONES DE COSAS SOBRE LAS QUE HABLAR"
2180 DATA "ATIENDAME AHORA. RELAJESE...Y HABLEMOS
     UN POCO SOBRE SU SITUACION"
2190 DATA "PUEDE SER"
2200 DATA "PARECE VACILAR UN POCO"
2210 DATA "PARECE BASTANTE INDECISO"
2220 DATA "EN QUE OTRAS CIRCUNSTANCIAS SE MUESTRA
     TAN INDECISO?"
2230 DATA "NO"
2240 DATA "POR QUE SE MUESTRA TAN NEGATIVO SOBRE
     ESO?"
2250 DATA "ESTA USTED DICIENDO QUE ESO NO ES SER
     NEGATIVO?"
2260 DATA "ES USTED BASTANTE IMPULSIVO. QUE LE SU
     GIERE ESO?"
2270 DATA "SIEMPRE"
2280 DATA "POR FAVOR, DEME UN EJEMPLO CONCRETO"
2290 DATA "NO LE PARECE QUE 'SIEMPRE' ES UN POCO
     EXAGERADO?"
```

```
2300 DATA "CUANDO?"
2310 DATA "CREO"
2320 DATA "DUDA*"
2330 DATA "PIENSA REALMENTE ESO?"
2340 DATA "PERO NO ESTA SEGURO*"
2350 DATA "AMIGO"
2360 DATA "POR QUE TOCA AHORA EL TEMA DE LOS AMI
     GOS?"
2370 DATA "CUAL ES SU MEJOR RECUERDO DE UN AMIGO?"
2380 DATA "POR FAVOR HABLEME MAS SOBRE SUS AMISTA
     DES..."
2390 DATA "AMIGOS"
2400 DATA "EN QUE MODO LE AFECTAN LAS REACCIONES
     DE SUS AMIGOS?"
2410 DATA "QUE LE HIZO COMENZAR A HABLAR DE LOS
     AMIGOS, PRECISAMENTE AHORA?"
2420 DATA "DE QUE MODO LE ENGANAN SUS AMIGOS?"
2430 DATA "SI"
2440 DATA "QUE TIPO DE MENTIRAS LE MOLESTAN?"
2450 DATA "ES USTED BASTANTE POSITIVO"
2460 DATA "PARECE MUY SEGURO DE ELLO"
2470 DATA "ORDENADOR"
2480 DATA "QUE ES LO QUE SIENTE SENTADO AHI Y HA
     BLANDOME COMO A UNA PERSONA?"
2490 DATA "LO PIENSA SOBRE MI EN PARTICULAR?"
2500 DATA "QUE ASPECTOS DE LOS ORDENADORES LE INTE
     RESAN MAS?"
2510 DATA "ESTA"
2520 DATA "PIENSA QUE ESTAR*"
2530 DATA "EN QUE CIRCUNSTANCIAS ESTARIA*"
2540 DATA "BIEN PODRIA SER QUE*"
2550 DATA "ES"
2560 DATA "QUE GRADO DE CERTEZA DEPOSITARIA EN
     ELLO SIENDO*"
2570 DATA "ESTA USTED SEGURO DE QUE ES*"
2580 DATA "QUE SENSACIONES EXPERIMENTARIA SI LE DI
     JERA QUE, PROBABLEMENTE NO ES*"
2590 DATA "PUEDE"
2600 DATA "QUE LE HACE PENSAR QUE NO PUEDO*"
2610 DATA "NO PIENSA QUE PUEDO*"
2620 DATA "QUIZAS LE GUSTARIA SER CAPAZ DE*"
2630 DATA "PUEDO"
2640 DATA "TAL VEZ NO QUIERA*"
2650 DATA "LE GUSTARIA SER CAPAZ DE*"
2660 DATA "LO DUDO"
2670 DATA "SOY"
2680 DATA "POR QUE PIENSA QUE SOY*"
2690 DATA "LE COMPLACE CREER QUE SOY*"
2700 DATA "TAL VEZ LE GUSTARIA SER*"
2710 DATA "ESTA"
2720 DATA "POR QUE PIENSA QUE ESTOY*"
2730 DATA "LE COMPLACE CREER QUE ESTOY*"
2740 DATA "POR QUE DICE QUE ESTOY*"
2750 DATA "YO NO"
2760 DATA "REALMENTE NO*"
2770 DATA "POR QUE NO*"
2780 DATA "LE GUSTARIA SER CAPAZ DE*"
```

```
2790 DATA "SIENTO"
2800 DATA "HABLEME MAS SOBRE ESOS SENTIMIENTOS"
2810 DATA "SIENTE A MENUDO*"
2820 DATA "DISFRUTA SINTIENDOSE*"
2830 DATA "SIENTO"
2840 DATA "VAMOS A EXAMINAR ESA AFIRMACION MAS DE
     TENIDAMENTE"
2850 DATA "SE SIENTE NORMALMENTE DE ESE MODO?"
2860 DATA "QUE EMOCIONES LE DESPIERTAN ESOS SENTI
     MIENTOS?"
2870 DATA "TENGO"
2880 DATA "POR QUE ME DICE QUE TIENE*"
2890 DATA "PARA MI ESTA CLARO QUE TIENE*"
2900 DATA "COMO PODRIA AYUDARLE CON*"
2910 DATA "PODRIA"
2920 DATA "PODRIA EXPLICARME POR QUE SERIA*"
2930 DATA "A QUIEN MAS LE HA DICHO QUE USTED PO
     DRIA*"
2940 DATA "QUE SEGURIDAD TIENE DE QUE PODRIA"
2950 DATA "HAY"
2960 DATA "POR SUPUESTO QUE HAY*"
2970 DATA "ES COMO SI HUBIERA*"
2980 DATA "LE GUSTARIA QUE HUBIERA*"
2990 DATA "MI"
3000 DATA "SU"
3010 DATA "YA VEO, SU*"
3020 DATA "QUE SIGNIFICA PARA USTED EL QUE SU*"
3030 DATA "USTED"
3040 DATA "ESTA SESION ES PARA AYUDARLE...NO PARA
     QUE ANDEMOS DISCUTIENDO"
3050 DATA "QUE LE IMPULSO A DECIR ESO SOBRE MI?"
3060 DATA "RECUERDE QUE TOMO NOTA DE TODO LO QUE
     ESTA DICIENDO, PARA PODER ARREGLAR SU SITUA
     CION"
3070 DATA *, *, *, *
```



9
Traducción por máquinas

Cuando pensamos en las posibilidades que surgen del hecho de que las máquinas sean capaces de comprender y procesar en una lengua natural, no parece una extravagancia el imaginar que éstas mismas podrían resultarnos una gran ayuda a la hora de traducir de una lengua natural a otra. Esta esperanza no es algo nuevo entre nosotros, ya que la investigación en este campo se inició por los años cincuenta y desde entonces se han hecho grandes progresos.

Son ahora más de 20 los sistemas de traducción mediante máquinas que funcionan por el mundo. Pero, contrariamente a la opinión popular, estos sistemas no funcionan sobre una base del tipo INTRODUZCA EL DOCUMENTO EN INGLES POR ESTA RANURA Y RECOJA LA VERSION EN ESPAÑOL POR ESTA OTRA. La traducción mediante máquinas es mucho más sutil y complicada. En realidad, hay varios subcampos dentro del dominio general de la traducción con ordenadores.

Su uso

Aunque en los primeros momentos del desarrollo de los sistemas de traducción se aceptó la idea (posiblemente sin recapacitar sobre ello todo lo necesario) de que los traductores humanos acabarían siendo innecesarios, los investigadores confirman ahora que, por el momento (y en un futuro inmediato), el papel de los traductores humanos sigue siendo vital. Los especialistas de este campo de investigación se refieren últimamente al concepto de "pre-traducción mediante máquinas" considerando los documentos traducidos por los ordenadores como simples borradores sobre los que trabajar.

Hay varios modos diferentes de concebir la traducción mediante ordenadores, todos ellos en uso actualmente. Entre éstos podemos incluir los sistemas construidos con la idea de traducir, ya sea documentos escritos en una versión reducida o limitada de una lengua natural, ya sea textos que antes de ser introducidos en la máquina han pasado

por un proceso de elaboración, de modo que la máquina pueda comprenderlos y manejarlos con más facilidad. Xerox tiene un sistema de este tipo, llamado SYSTRAN. Observaremos, cuando llegue el momento, algunas de las salidas producidas por este sistema (en traducciones de documentos para la CEE).

Otra manera de concebir la traducción con máquinas es aquella en la que el usuario puede modificar el sistema de acuerdo con sus propias necesidades, proporcionándole un vocabulario que se adecue a la especialidad sobre la que vaya a traducir el ordenador. Tal sistema, llamado CULT, se utiliza actualmente en Hong Kong, donde traduce revistas matemáticas chinas. La impresión de la máquina se encuaderna tal cual (sin corrección posterior) y se vende en todo el mundo.

Cuando tú y yo, lector, como legos en la materia, pensamos en la traducción mediante ordenadores, lo más probable es que nos imaginemos unas máquinas perfectas en las que uno introduce una entrada en inglés y obtiene una salida en español. Y éste es, en realidad, el objetivo final de todos aquellos que están investigando en este campo, aunque por el momento están todavía muy lejos de alcanzarlo. No obstante, el sistema SYSTRAN —que, como acabamos de mencionar, trabaja ya sea con documentos escritos en "sub-inglés", ya sea con documentos que han sido previamente preparados para este fin— puede usarse a modo de *freelance,* abordando cualquier documento que se le introduzca. Los resultados obtenidos varían de un documento a otro.

Muchos documentos pasan por una fase de preparación antes de ser introducidos en la máquina para que ésta los traduzca. Con este proceso se intenta suprimir las potenciales ambigüedades y otros aspectos del texto que pudieran confundir a la máquina. Muchos de los documentos traducidos por la máquina precisan una elaboración posterior (la mayoría, de hecho). En esta fase se comprueba si hay errores y se corrige la sintaxis.

Algunos documentos no tienen que pasar por esta fase de corrección posterior. En ciertos casos, el borrador que sale directamente de la máquina puede ser suficiente.

La traducción mediante máquinas puede llevarse también a cabo con la ayuda de un traductor humano que interviene mientras se está llevando a cabo la traducción.

Como puede verse, el papel del humano sigue siendo vital en el proceso de traducción. Y no hay nada que nos indique que esto vaya a cambiar en un futuro próximo. Las máquinas pueden hacer borradores de las traducciones, pero la corrección y la redacción final por parte del humano sigue siendo necesaria.

Examinemos un ejemplo genuino de una traducción realizada por ordenadores. Se trata de un documento de la CEE traducido del francés al inglés por el sistema SYSTRAN en 1981.

Este es el inicio del documento en francés:

```
Application de la micrologique au
controle des operations de production

But de la recherche:

Perfectionner les appareillages existants
de sorte que les preposes soient
debarasses des taches dans lesquelles
leur jugement n'intervient pas.

Application au central de telesurveillance
d'engins sur pneus
```

La máquina respondió con esta traducción:

```
Application of micrological to the
control of the production operations

Aim of the research:

To improve existing equipments so that
the officials debarasses tasks in which
their judgement does not intervene

Application to the exchange of
telesurveillance of equipment on tyres
```

Aunque no es del todo exacta, deja ver una gran parte del significado. El *debarasses* de la traducción inglesa se debe, de hecho, a una falta en el francés del original (debería decir *debarrasses*, en cuyo caso, presumiblemente, la máquina hubiera comprendido).

Tras haber sido corregido y redactado por un traductor humano, el documento quedó con esta forma:

```
Application of micrology to the
monitoring of production operations

Aim of the research:

To perfect existing apparatus so that
staff can be relieved of tasks where no
judgement is required.

Application to the remote monitoring
station for trackless vehicles.
```

Para mí es fascinante seguir el modo en que se ha traducido el documento. Salvo la última línea, la versión definitiva del texto inglés no es muy diferente de la salida original producida por SYSTRAN.

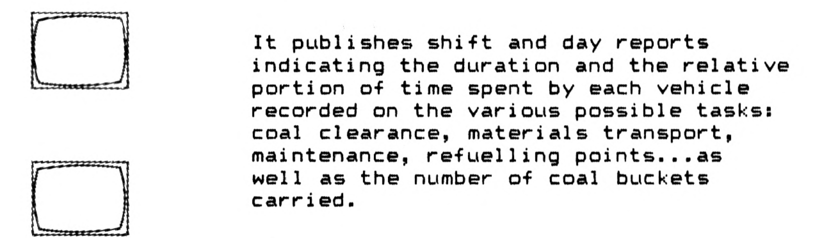
No todo el documento se tradujo igual de bien. El traductor-corrector humano hubo de corregir enormemente el resto del texto, reduciéndolo a la sombra de lo que fue.

Este es el que imprimió la máquina:

```
It publishes station and day reports
indicating the duration and the
importance relative of the periods
devoted by each instrument supervised to
the various possible activities:
evacuation of the products, transport of
equipment, maintenance, station service
...as well as the number of evacuated
coal cups.
```

Es el tipo de texto que revela, tal como es, la naturaleza de la traducción mediante máquinas, con frases como *the importance relative of the periods,* que muestran claramente el origen francés del texto.

Tras la corrección y redacción final, el texto quedó así:



```
It publishes shift and day reports
indicating the duration and the relative
portion of time spent by each vehicle
recorded on the various possible tasks:
coal clearance, materials transport,
maintenance, refuelling points...as
well as the number of coal buckets
carried.
```

Finalmente, antes de pasar a crear nuestro propio programa de traducción, es interesante observar que la inmensa mayoría de los documentos traducidos hoy en día mediante máquinas son textos no literarios. La traducción de obras de literatura constituye un campo totalmente diferente y, en lo que se refiere a la traducción mediante máquinas, está todavía en pañales.

"Spanglish"

Este programa, en el que se utiliza un vocabulario ideado por Jeremy Ruston y que está basado en una idea suya, acepta una entrada en español y produce una extraña mezcla políglota de inglés y español, en las que las palabras más fáciles y obvias están traducidas a español y las más difíciles dejadas en inglés.

El programa que ofrecemos aquí no intenta ser un programa serio. No obstante, sí que indica algunos de los problemas inherentes a la traducción mediante máquinas. Tomado de un modo más serio, con un vocabulario mucho más extenso, podría usarse para producir el borrador de un texto traducido del español al inglés (o a la inversa, cambiando sólo dos variables) que posteriormente tendría que ser extensamente corregido y redactado. Si el programa se usara en un campo con un vocabulario especializado, podría prestar un servicio relativamente bueno, aunque no sería capaz de hacer ningún análisis que garantizara (como la exigencia del género en español) que todas las partes de la frase son correctas.

Posiblemente habrá quien piense, al leer algunas de las salidas del programa, que soy muy poco realista cuando afirmo que éste, ampliándole el vocabulario, podría servir para fines más serios. Sin embargo, si uno se para a pensar sobre ello, se dará cuenta de que en su forma presente apenas se saca partido a todo el potencial que ofrece el programa.

Aquí tenemos algunas salidas de TRADUCTOR:

```
ESCRIBE LO QUE QUIERAS TRADUCIR? Hola Carlos buenos dias
   -->HELLO CARLOS GOOD MORNING

ESCRIBE LO QUE QUIERAS TRADUCIR? Ellos tienen un perro
   -->THEY HAVE A DOG
```

```
ESCRIBE LO QUE QUIERAS TRADUCIR? Maria tiene cigarrillos
  -->MARIA HAS CIGARRETES

ESCRIBE LO QUE QUIERAS TRADUCIR? El filete tiene mostaza
  -->THE STEAK HAS MUSTARD  .

ESCRIBE LO QUE QUIERAS TRADUCIR? Juan ha bebido un guiski
                                 con agua
  -->JUAN HAS DRUNK A WHISKY
     WITH WATER
```

Como puede verse, he introducido en español aquellas claves que constan en el vocabulario, extremadamente limitado, del programa:

```
ESCRIBE LO QUE QUIERAS TRADUCIR? Por favor la chica es mia
  -->PLEASE THE GIRL IS MINE

ESCRIBE LO QUE QUIERAS TRADUCIR? Mi padre ha estado en el mar
  -->MY FATHER HAS BEEN IN THE SEA

ESCRIBE LO QUE QUIERAS TRADUCIR? El sol esta caliente y
                                 la luna esta fria

  -->THE SUN IS HOT AND
     THE MOON IS COLD

ESCRIBE LO QUE QUIERAS TRADUCIR? El chico tiene una casa
                                 con musica
  -->THE BOY HAS A HOUSE
     WITH MUSIC

ESCRIBE LO QUE QUIERAS TRADUCIR? Pedro ha comido una hamburguesa
                                 con patatas fritas y ha bebido
                                 vino
  -->PETER HAS EATEN A HAMBURGER
     WITH FRITES AND HAS DRUNK
     WINE
```

Estructura del programa

Es muy fácil de seguir. Empieza (como siempre) con una llamada a una subrutina situada al final del programa que inicializa las variables.

```
 10 REM TRADUCTOR
 20 GOSUB 400: REM INICIALIZACION

400 REM INICIALIZACION
410 CLS
420 DEFINT A - Z
430 DIM E$(110): REM ALMACENA LAS PALABRAS EN
    ESPANOL
440 DIM I$(110): REM ALMACENA LAS PALABRAS EN
    INGLES
```

```
450 CCONTADOR = 0
460 CCONTADOR = CCONTADOR + 1
470 READ E$(CCONTADOR), F$(CCONTADOR)
480 IF F$(CCONTADOR) <> "*" THEN 9105
490 RETURN
500 REM ** DATOS **
510 DATA YO, I, TU, YOU, USTED, YOU, ELLA, SHE,
    ELLO, IT, NOSOTROS, WE, VOSOTROS, YOU, ELLOS,
    THEY
520 DATA EL, THE, LA, THE, LOS, THE, LAS, THE,
    UN, A, UNA, A
530 DATA MIO, MINE, TUYO, YOURS, NUESTRO, OUR,
    VUESTRO, YOUR
540 DATA SOY, AM, ERES, ARE, ES, IS, SOMOS, ARE,
    SOIS, ARE, SON, ARE
550 DATA TENGO, HAVE, TIENES, HAVE, TIENE, HAS,
    TENEMOS, HAVE, TENEIS, HAVE, TIENEN, HAVE
560 DATA HE, HAVE, HAS, HAVE, HA, HAS, HEMOS,
    HAVE, HABEIS, HAVE, HAN, HAVE
570 DATA COMIDO, EATEN, BEBIDO, DRUNK, TENIDO,
    HAD, SIDO, BEEN
580 DATA Y, AND, E, AND, O, OR, U, OR, SI, YES,
    NO, NO, DE, OF, CON, WITH, EN, IN
590 DATA BRAZO, ARM, PIERNA, LEG, OJO, EYE,
    CABEZA, HEAD, DERECHA, RIGHT, DERECHO, RIGHT,
    IZQUIERDA, LEFT, IZQUIERDO, LEFT
600 DATA GINEBRA, GIN, VINO, WINE, GUISKY, WHISKY
    , CERVEZA, BEER, AGUA, WATER
610 DATA PESCADO, FISH, POLLO, CHICKEN, PATO,
    DUCK, MOSTAZA, MUSTARD, SAL, SALT
620 DATA PATATAS FRITAS, FRITES, FILETE, STEAK,
    SOPA, SOUP, HAMBURGUESA, HAMBURGER
630 DATA CALIENTE, HOT, FRIO, COLD, TODO EL MUNDO
    , EVERYBODY
640 DATA MUSICA, MUSIC, CHICO, BOY, CHICA, GIRL,
    HOMBRE, MAN, MUJER, WOMAN, POR FAVOR, PLEASE
650 DATA PERRO, DOG, GATO, CAT, PAJARO, BIRD,
    CERDO, PIG, OVEJA, SHEEP
660 DATA HOLA, HELLO, BUENOS, GOOD, DIAS, MORNING
    , NOCHES, NIGHT, ADIOS, GOODBYE
670 DATA PUERTA, DOOR, CASA, HOUSE, AMOR, LOVE,
    SOL, SUN, LUNA, MOON, POLICIA, POLICEMAN
680 DATA CANCION, SONG, AMIGOS, FRIENDS, MAR, SEA
    , CIGARRILLOS, CIGARETTES
690 DATA *,*
```

En esta subrutina, E$ se utiliza para contener el texto español e I$ para contener su equivalente en inglés. El equivalente inglés de E$(4) (aquí) es I$(4) (here) y así sucesivamente, lo que hace su uso muy sencillo.

La variable CONTADOR cuenta el número de palabras introducidas en el sistema. Las matrices han sido dimensionadas de modo que puedan contener más palabras de las que tienen ahora, a fin de que cada cual pueda añadir las suyas propias.

Al volver de la rutina de inicialización, el programa acepta la entrada del usuario

(línea 30) y después comprueba si hay una cadena vacía (es decir, verifica que el usuario no ha pulsado simplemente RETURN sin hacer ninguna entrada). Si encuentra que la entrada, A$, está vacía, el programa termina.

```
30 INPUT "ESCRIBE LO QUE QUIERAS TRADUCIR"; A$:
   REM ACEPTA LA FRASE DEL USUARIO
40 IF A$ = "" THEN PRINT "FIN DEL PROGRAMA": END
50 GOSUB 290: REM TRANSFORMA EN MAYUSCULAS
60 GOSUB 100: REM TRADUCCION
70 GOTO 30
```

La línea 50 envía el texto a la subrutina que empieza en la línea 290, la cual pone en mayúsculas la entrada el usuario y después la subrutina que empieza en la línea 100 se encarga de hacer la traducción propiamente dicha. La línea 70 hace retroceder el programa a la 30, en donde acepta más entrada del usuario.

Esta es la rutina que pone en mayúsculas la entrada del usuario:

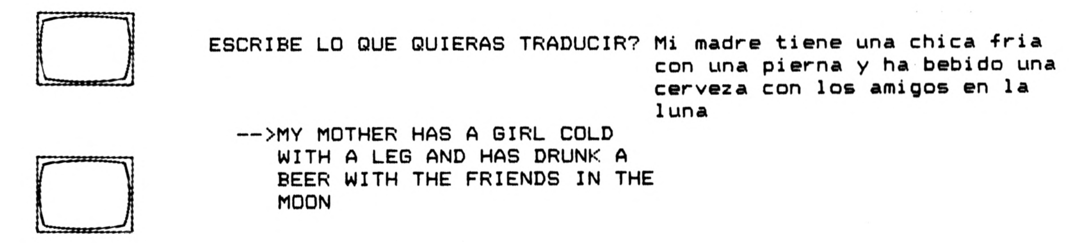
```
290 REM TRANSFORMA EN MAYUSCULAS
300 A$ = " " + A$ + " "
310 B$ = ""
320 L = LEN(A$)
330 FOR J = 1 TO L
340 K = ASC(MID$(A$, J, 1))
350 IF K > 96 AND K < 124 THEN K = K - 32
360 B$ = B$ + CHR$(K)
370 NEXT J
380 RETURN
```

Sencillamente recorre el texto, elemento por elemento, convirtiendo todos los caracteres cuyo código ASCII esté entre 97 y 123 (es decir, las letras minúsculas) en su equivalente en mayúsculas (restando 32). Una vez que se ha transformado en mayúsculas toda la entrada, empieza la traducción propiamente dicha:

```
100 REM TRADUCCION
110 PRINT TAB(2); "-->";
120 K = 0
130 K = K + 1
140 IF K = L THEN PRINT: PRINT: RETURN
150 IF MID$(B$, K, 1) = " " THEN 170
160 GOTO 130
170 X = K + 1
180 Y = 0
190 Y = Y + 1
```

```
200 IF MID$(B$, (X + Y), 1) = " " THEN Q$ =
    MID$(B$, X, Y) : GOTO 220
210 GOTO 190
220 M = 0
230 M = M + 1
240 IF Q$ = E$(M) THEN PRINT I$(M); " "; : GOTO
    270
250 IF M < CCONTADOR THEN 230
260 PRINT Q$; " ";
270 GOTO 130
```

El programa recorre el texto, buscando el espacio que indica el inicio de una nueva palabra (la palabra, por supuesto, empieza después del espacio; esta es la razón por la que añadimos —en la línea 30— un espacio al final de cada entrada, de tal modo que el programa no pasará por alto la primera y la última palabra de cada frase). Una vez que lo encuentra (línea 150) va a la rutina que empieza en la línea 170, la cual continúa buscando el siguiente espacio, de modo que puede aislar la palabra entera. Acto seguido se limita a recorrer el vocabulario (líneas 220 a 270) hasta que encuentra su pareja.

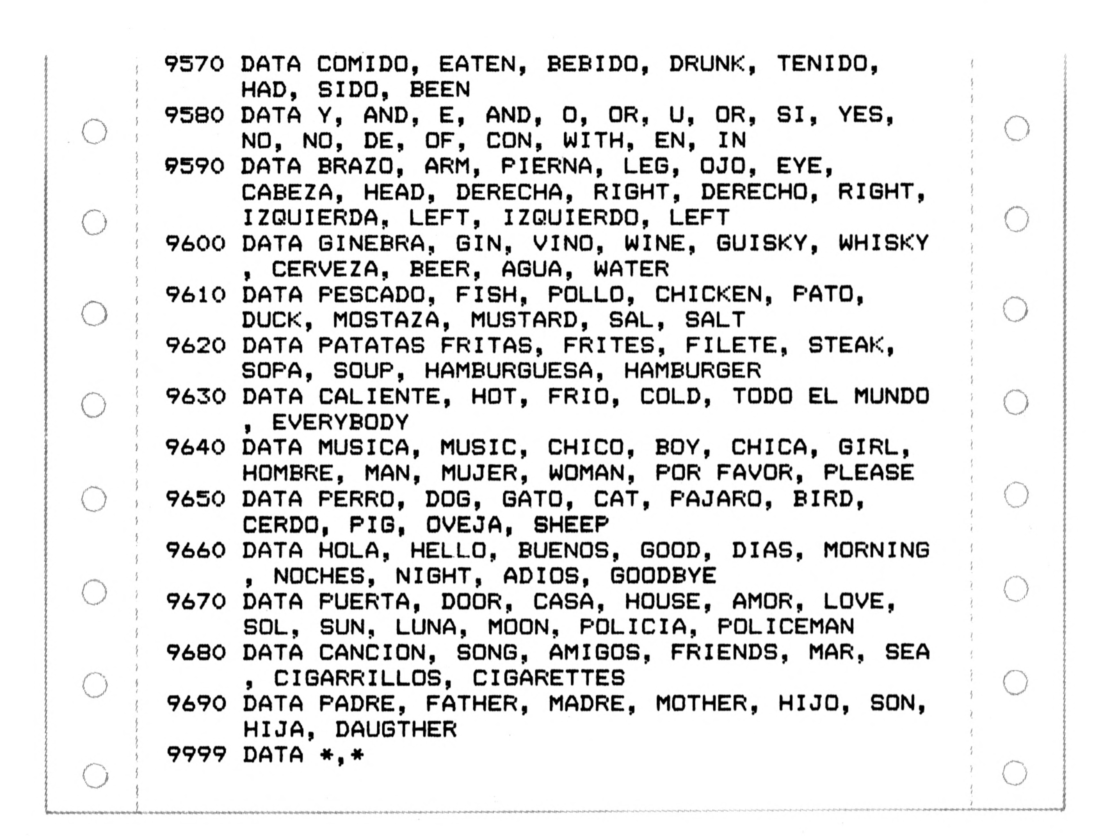


Si encuentra su equivalente, imprime la palabra inglesa en vez de la española y el programa vuelve a la línea 130 para continuar la búsqueda. Observemos que una vez que ha encontrado un equivalente, el programa vuelve a la línea 130. No pierde tiempo buscando en el resto del vocabulario. Esto significa que las palabras situadas al principio de la lista serán traducidas más deprisa que las situadas al final. Esta es la razón por la que hemos puesto al inicio de la lista las palabras más usuales (como EL, MI, SOY).

Ha llegado el momento de que cada uno experimentemos por nuestra cuenta (para usuarios del ZX Spectrum, véase Apéndices):

```
  10 REM TRADUCTOR
  20 GOSUB 9000: REM INICIALIZACION
  30 INPUT "ESCRIBE LO QUE QUIERAS TRADUCIR"; A$:
     REM ACEPTA LA FRASE DEL USUARIO
  35 IF A$ = "" THEN PRINT "FIN DEL PROGRAMA": END
  40 GOSUB 8000: REM TRANSFORMA EN MAYUSCULAS
  50 GOSUB 1000: REM TRADUCCION
  60 GOTO 30
 980 END
 999 REM **********
1000 REM TRADUCCION
1002 PRINT TAB(2); "-->";
```

```
1005 K = 0
1010 K = K + 1
1015 IF K = L THEN PRINT: PRINT: RETURN
1020 IF MID$(B$, K, 1) = " " THEN 1200
1030 GOTO 1010
1200 X = K + 1
1210 Y = 0
1220 Y = Y + 1
1230 IF MID$(B$, (X + Y), 1) = " " THEN Q$ =
     MID$(B$, X, Y) : GOTO 1240
1235 GOTO 1220
1240 M = 0
1250 M = M + 1
1260 IF Q$ = E$(M) THEN PRINT I$(M); " "; : GOTO
     1280
1265 IF M < CCONTADOR THEN 1250
1270 PRINT Q$; " ";
1280 GOTO 1010
7999 REM ************************
8000 REM TRANSFORMA EN MAYUSCULAS
8005 A$ = " " + A$ + " "
8010 B$ = ""
8020 L = LEN(A$)
8030 FOR J = 1 TO L
8040 K = ASC(MID$(A$, J, 1))
8050 IF K > 96 AND K < 124 THEN K = K - 32
8060 B$ = B$ + CHR$(K)
8070 NEXT J
8600 RETURN
8999 REM **************
9000 REM INICIALIZACION
9005 CLS
9006 DEFINT A - Z
9010 DIM E$(110): REM ALMACENA LAS PALABRAS EN
     ESPANOL
9020 DIM I$(110): REM ALMACENA LAS PALABRAS EN
     INGLES
9100 CCONTADOR = 0
9105 CCONTADOR = CCONTADOR + 1
9110 READ E$(CCONTADOR), F$(CCONTADOR)
9120 IF F$(CCONTADOR) <> "*" THEN 9105
9490 RETURN
9500 REM ** DATOS **
9510 DATA YO, I, TU, YOU, USTED, YOU, ELLA, SHE,
     ELLO, IT, NOSOTROS, WE, VOSOTROS, YOU, ELLOS,
     THEY
9520 DATA EL, THE, LA, THE, LOS, THE, LAS, THE,
     UN, A, UNA, A
9530 DATA MIO, MINE, TUYO, YOURS, NUESTRO, OUR,
     VUESTRO, YOUR
9540 DATA SOY, AM, ERES, ARE, ES, IS, SOMOS, ARE,
     SOIS, ARE, SON, ARE
9550 DATA TENGO, HAVE, TIENES, HAVE, TIENE, HAS,
     TENEMOS, HAVE, TENEIS, HAVE, TIENEN, HAVE
9560 DATA HE, HAVE, HAS, HAVE, HA, HAS, HEMOS,
     HAVE, HABEIS, HAVE, HAN, HAVE
```



```
9570 DATA COMIDO, EATEN, BEBIDO, DRUNK, TENIDO,
     HAD, SIDO, BEEN
9580 DATA Y, AND, E, AND, O, OR, U, OR, SI, YES,
     NO, NO, DE, OF, CON, WITH, EN, IN
9590 DATA BRAZO, ARM, PIERNA, LEG, OJO, EYE,
     CABEZA, HEAD, DERECHA, RIGHT, DERECHO, RIGHT,
     IZQUIERDA, LEFT, IZQUIERDO, LEFT
9600 DATA GINEBRA, GIN, VINO, WINE, GUISKY, WHISKY
     , CERVEZA, BEER, AGUA, WATER
9610 DATA PESCADO, FISH, POLLO, CHICKEN, PATO,
     DUCK, MOSTAZA, MUSTARD, SAL, SALT
9620 DATA PATATAS FRITAS, FRITES, FILETE, STEAK,
     SOPA, SOUP, HAMBURGUESA, HAMBURGER
9630 DATA CALIENTE, HOT, FRIO, COLD, TODO EL MUNDO
     , EVERYBODY
9640 DATA MUSICA, MUSIC, CHICO, BOY, CHICA, GIRL,
     HOMBRE, MAN, MUJER, WOMAN, POR FAVOR, PLEASE
9650 DATA PERRO, DOG, GATO, CAT, PAJARO, BIRD,
     CERDO, PIG, OVEJA, SHEEP
9660 DATA HOLA, HELLO, BUENOS, GOOD, DIAS, MORNING
     , NOCHES, NIGHT, ADIOS, GOODBYE
9670 DATA PUERTA, DOOR, CASA, HOUSE, AMOR, LOVE,
     SOL, SUN, LUNA, MOON, POLICIA, POLICEMAN
9680 DATA CANCION, SONG, AMIGOS, FRIENDS, MAR, SEA
     , CIGARRILLOS, CIGARETTES
9690 DATA PADRE, FATHER, MADRE, MOTHER, HIJO, SON,
     HIJA, DAUGTHER
9999 DATA *,*
```



10 HANSHAN

El último programa de esta sección dedicada al manejo de una lengua natural crea poemas al azar. Es un programa bastante trivial y que apenas nos ofrece evidencia alguna de inteligencia artificial.

Sin embargo, imagínese el lector leyendo un libro de este tipo hace treinta años. El autor hace un comentario intrascendente sobre un mecanismo barato que escribe poesía de modo automático y después lo rechaza diciendo que carece de importancia. Hace treinta años esto hubiera sido algo extraordinario. Y, en realidad, cuando nos paramos a pensar sobre ello, sigue siéndolo. Nos hemos acostumbrado tanto al milagro que ahora pasa por nuestro lado y no lo vemos.

Así pues, con esta idea *in mente,* nos volvemos hacia HANSHAN para crear unos cuantos poemas. El programa recibe su nombre del poeta chino Han-Shan, que vivió durante los últimos años del siglo VIII y primeros del XI. Tras romper con su vida familiar, vagabundeó durante muchos años para terminar haciéndose ermitaño en las montañas de las cuales toma a su nombre.

Todas las frases usadas en el DATA de este programa están sacadas del libro *Chinese Poems* (Arthur Waley, Unwin Papersbacks, Londres, 1982):

```
370 REM ** PALABRAS SUELTAS **
380 DATA ESCABULLENDOSE, PISANDO, CONTEMPLANDO,
         MARCHITO, ESCULPIDO
390 DATA ESCONDIDO, CONFINADO, ATORMENTADO,
         DOBLANDO, RETORCIENDO
400 DATA GOLPEADO, COLGANDO, ENVOLVIENDO,
         TRANSPARENTE, CANSADO
410 DATA HACIA LA TIERRA, CASCADA, SACRIFICADOR,
         RESBALADIZO, EN PEDAZOS
```

```
420 REM ** FRASES CORTAS **
430 DATA EN LA FRIA CORRIENTE
440 DATA AJENO A UN MUNDO DE BELLEZA
450 DATA OLAS TRANQUILAS
460 DATA FUERA, DESDE EL ABISMO
470 DATA "SOMBRIO, SOMBRIO"
480 DATA EN LA NEGRURA DE LA OSCURIDAD
490 DATA COGI TUS POEMAS
500 DATA APAGUE LA LAMPARA
510 DATA SE ME CIERRAN LOS OJOS
520 DATA AQUELLOS QUE ESTAN A LA IZQUIERDA
530 DATA LOS HOMBRES DE CIENCIA
540 DATA LOS HOMBRES DE ACCION
550 DATA ME APRESURO
560 DATA POR QUE DESPERDICIARIAS
570 DATA CUANDO NOS ENCONTRAREMOS DE NUEVO?
580 DATA DURMIENDO UN POCO
590 DATA Y CON MUCHA PENA
600 DATA POR ESTOS POCOS PASOS
610 DATA AHORA, AL ANOCHECER
620 DATA HICE PROVECHOSAMENTE
```

El programa selecciona uno de los tres modelos a partir de los cuales crea poemas parecidos a los haikus (el haiku, por supuesto, es una forma poética japonesa, pero al programa le tiene sin cuidado que se pueda producir un antagonismo entre las frases chinas y la forma en que él las coloca —que es más bien el tipo de estructura del haiku japonés):

```
 30 REM ELECCION DEL PATRON
 40 R = INT(RND(1) * 3) + 1
 50 ON R GOSUB 90, 140, 190
 60 FOR T = 1 TO 1000: NEXT T
 70 PRINT: PRINT: PRINT
 80 GOTO 40
 90 REM ** PATRON UNO **
100 PRINT W$(INT(RND(1) * 20 + 1)); "...";
          W$(INT(RND(1) * 20 + 1))
110 PRINT TAB(5); "..."; W$(INT(RND(1) * 20 + 1))
120 PRINT TAB(8); S$(INT(RND(1) * 20 + 1))
130 RETURN
140 REM ** PATRON DOS **
150 PRINT S$(INT(RND(1) * 20 + 1))
160 PRINT TAB(3); S$(INT(RND(1) * 20 + 1)); "..."
170 PRINT TAB(6); S$(INT(RND(1) * 20 + 1))
180 RETURN
190 REM ** PATRON TRES **
200 PRINT TAB(3); W$(INT(RND(1) * 20 + 1))
210 PRINT S$(INT(RND(1) * 20 + 1))
220 PRINT TAB(3); W$(INT(RND(1) * 20 + 1)); ", ";
          S$(INT(RND(1) * 20 + 1)); ", ";
          S$(INT(RND(1)) * 20 + 1)
230 RETURN
```

Algunos de los poemas producidos por HANSHAN tienen bastante mérito:

```
CUANDO NOS ENCONTRAREMOS DE NUEVO?
   LOS HOMBRES DE ACCION...
      OLAS TRANQUILAS
```

```
APAGUE LA LAMPARA
   ME APRESURO...
      POR ESTOS POCOS PASOS
```

```
COLGANDO...CONTEMPLANDO
     ...DOBLANDO
        DESDE EL ABISMO
```

```
HACIA LA TIERRA...ESCONDIDO
     ...ESCULPIDO
        EN LA NEGRURA DE LA OSCURIDAD
```

```
   ESCABULLENDOSE
DESDE EL ABISMO
   CANSADO, COGI TUS POEMAS, EN LA FRIA CORRIENTE
```

```
ME APRESURO
   LOS HOMBRES DE CIENCIA...
      FUERA
```

```
Y CON MUCHA PENA
   Y CON MUCHA PENA...
      AJENO A UN MUNDO DE BELLEZA
```

Este es el listado de HANSHAN que le permitirá al lector crear una secuencia de poemas casi infinita. Cada cual puede modificar las sentencias DATA a su gusto, con lo que conseguirá un programa (y una salida) más personal (para usuarios del ZX Spectrum, véase Apéndice):

```
 10 REM HANSHAN
 20 GOSUB 250: REM INICIALIZACION
 30 REM ELECCION DEL PATRON
 40 R = INT(RND(1) * 3) + 1
 50 ON R GOSUB 90, 140, 190
 60 FOR T = 1 TO 1000: NEXT T
 70 PRINT: PRINT: PRINT
 80 GOTO 40
 90 REM ** PATRON UNO **
100 PRINT W$(INT(RND(1) * 20 + 1)); "...";
          W$(INT(RND(1) * 20 + 1))
110 PRINT TAB(5); "..."; W$(INT(RND(1) * 20 + 1))
120 PRINT TAB(8); S$(INT(RND(1) * 20 + 1))
130 RETURN
140 REM ** PATRON DOS **
150 PRINT S$(INT(RND(1) * 20 + 1))
160 PRINT TAB(3); S$(INT(RND(1) * 20 + 1)); "..."
170 PRINT TAB(6); S$(INT(RND(1) * 20 + 1))
180 RETURN
```

```
190 REM ** PATRON TRES **
200 PRINT TAB(3); W$(INT(RND(1) * 20 + 1))
210 PRINT S$(INT(RND(1) * 20 + 1))
220 PRINT TAB(3); W$(INT(RND(1) * 20 + 1)); ", ";
          S$(INT(RND(1) * 20 + 1)); ", ";
          S$(INT(RND(1)) * 20 + 1)
230 RETURN
240 REM *************
250 REM INICIALIZACION
260 CLS
270 RANDOMIZE
280 DIM W$(20), S$(20)
290 FOR J = 1 TO 20
300 READ W$(J)
310 NEXT J
320 FOR J = 1 TO 20
330 READ S$(J)
340 NEXT J
350 RETURN
360 REM ** DATOS **
370 REM ** PALABRAS SUELTAS **
380 DATA ESCABULLENDOSE, PISANDO, CONTEMPLANDO,
         MARCHITO, ESCULPIDO
390 DATA ESCONDIDO, CONFINADO, ATORMENTADO,
         DOBLANDO, RETORCIENDO
400 DATA GOLPEADO, COLGANDO, ENVOLVIENDO,
         TRANSPARENTE, CANSADO
410 DATA HACIA LA TIERRA, CASCADA, SACRIFICADOR,
         RESBALADIZO, EN PEDAZOS
420 REM ** FRASES CORTAS **
430 DATA EN LA FRIA CORRIENTE
440 DATA AJENO A UN MUNDO DE BELLEZA
450 DATA OLAS TRANQUILAS
460 DATA FUERA, DESDE EL ABISMO
470 DATA "SOMBRIO, SOMBRIO"
480 DATA EN LA NEGRURA DE LA OSCURIDAD
490 DATA COGI TUS POEMAS
500 DATA APAGUE LA LAMPARA
510 DATA SE ME CIERRAN LOS OJOS
520 DATA AQUELLOS QUE ESTAN A LA IZQUIERDA
530 DATA LOS HOMBRES DE CIENCIA
540 DATA LOS HOMBRES DE ACCION
550 DATA ME APRESURO
560 DATA POR QUE DESPERDICIARIAS
570 DATA CUANDO NOS ENCONTRAREMOS DE NUEVO?
580 DATA DURMIENDO UN POCO
590 DATA Y CON MUCHA PENA
600 DATA POR ESTOS POCOS PASOS
610 DATA AHORA, AL ANOCHECER
620 DATA HICE PROVECHOSAMENTE
```



Sección cuarta: LA AYUDA

11

Sistemas expertos

En nuestro mundo, el número de expertos sobre un tema específico es limitado. Ya hablemos de un tema o de otro —desde la reparación de automóviles o las excavaciones de uranio al diagnóstico de las enfermedades humanas, pasando por la clasificación de las setas comestibles— el número de expertos de que dispongamos será siempre restringido.

Ahora bien, mientras que nadie se lamenta porque no haya en el mundo más expertos en la clasificación de setas, hay, sin embargo, zonas inmensas de nuestro planeta (en la mayoría de éste, en realidad) en las que escasean los médicos. Una de las ideas de los sistemas expertos es ''capturar'' en un ordenador la pericia de una persona experta en un área determinada del conocimiento, de tal modo que una persona no experta pueda aprovechar esa información.

Los sistemas expertos constituyen el campo de investigación de la inteligencia artificial en el que se han dado los avances más significativos. Es, asimismo, en este campo de investigación donde se están haciendo ya actualmente las contribuciones más prácticas y viables económicamente. Por otro lado, es un campo de la inteligencia artificial en el que nadie anda preocupándose con cuestiones relativas a si la máquina piensa o deja de pensar cuando muestra su capacidad o pericia en un tema determinado.

En su forma más simple, un sistema experto está formado por una serie de sentencias SI/ENTONCES (IF/THEN). Un sistema de diagnóstico podría ser tan simple como:

SI el paciente tose
Y recientemente se empapó hasta los huesos
Y luego estuvo expuesto durante una hora a un viento helado,
ENTONCES el paciente tiene un resfriado o una pulmonía.

Nadie hubiera necesitado, desde luego, un sistema experto para hacer semejante diagnóstico (se ha de observar, además, que yo no estoy en absoluto sugeriendo que el diagnóstico que acabo de dar tenga que ser necesariamente correcto). Un sistema experto se hace valer cuando se da una de las siguientes condiciones:

— la persona experta no está presente, pero sí que lo está su pericia o conocimiento del tema;
— ni siquiera la persona experta sabe con un 100 por 100 de certeza la relación causal entre las observaciones y los resultados. Esto podría suceder si un investigador médico fuera consciente de que los pacientes que contraen la enfermedad X suelen haber tenido contacto con los alimentos A y B y tienen un grupo sanguíneo C..., aunque no se haya descubierto un modo de relacionar A, B y C —salvo por el hecho de que suelen aparecer juntos—. En este caso, un programa experto debidamente programado podría predecir las posibilidades de que el individuo D contrajera la enfermedad, aun cuando no se supiera en qué porcentaje contribuyen los factores A, B y C. Mediante el estudio de un número suficiente de casos, el sistema experto podría no sólo concebir sus propias reglas para predecir si una persona podría o no contraer esa enfermedad, sino también explicar su razonamiento al médico humano.

En la sección del libro dedicada al aprendizaje y el razonamiento hablábamos de los ''circuitos lógicos'' y comentábamos el modo en que éstos tomaban decisiones de acuerdo con las reglas del Algebra Booleana.

Las ''matemáticas del razonamiento'' son muy importantes en la construcción de sistemas expertos. Sucede a menudo que la persona que está ''sacando'' la pericia de un ser humano con el fin de codificarla en la base de datos de un sistema experto (un poco más adelante veremos algunos de los sistemas que actualmente se están utilizando en todo el mundo) descubre que aquella persona experta en tal o cual tema no sabe, de hecho, qué mecanismo emplea para llegar a una conclusión o tomar una decisión.

Esto es una revelación tanto para el experto como para la persona que está creando la base de datos para el programa. En el libro *The Fith Generation Artificial Intelligence and Japan's Computer Challenge to the World* (Reading, Massachusetts, Feigenbaum, Edward A. and McCorduck, Pamela, 1983, pp. 85 y 86) leí la triste historia de un experto que puso toda su buena voluntad en explicar sus métodos a un ''ingeniero del conocimiento'' (que es como se denomina a todos aquellos que se dedican a extraer la pericia de otros para, una vez modificada, pasarla a un programa informático). Aquel experto estaba muy bien considerado (y pagado) debido a su capacidad en un tema concreto y al principio no podía creerse que toda su pericia pudiera reducirse, según le descubrió el ingeniero del conocimiento, a unos cuantos cientos de ''reglas de trabajo empíricas''. De la incredulidad, el experto pasó a la depresión y terminó abandonando su campo de trabajo. Un hombre destrozado.

Las máquinas toman decisiones basadas en sus reglas internas. Estas son —como vimos en la explicación de los programas de aprendizaje y razonamiento— relativamente simples. El razonamiento elemental lógico se reduce a unas cuantas reglas bien expresadas.

Vimos que la máquina podía expresar y resolver silogismos debido a que éstos tomaban la siguiente forma:

A es C
C es B
Luego, A es B.

La esperanza de poder reducir el razonamiento a un proceso mecánico no es nada nuevo entre nosotros. Ya en 1677, en el prefacio a su libro *Las ciencias generales,* G. Leibniz escribía:

> *Si uno pudiera encontrar las características o signos apropiados para expresar todos nuestros pensamientos tan clara y exactamente como la aritmética expresa los números o la geometría analítica, las líneas, podríamos realizar en todos los campos del conocimiento, tanto en cuanto éstos puedan someterse al razonamiento, lo que se hace en el campo de la aritmética y la geometría...*
>
> *Es más, podríamos convencer al mundo de lo que hemos encontrado o concluido, puesto que sería muy fácil verificar el cálculo... si alguien dudara de mis resultados, bastaría con que le dijera: "Calculémoslo, señor", y tomando pluma y papel, enseguida zanjaríamos la cuestión.*

Ahora podemos sustituir la pluma y el papel por sílice y encontrar la solución de algunos problemas que, de no ser así, sobrepasarían las capacidades de la mayoría de nosotros (como, por ejemplo, la capacidad para predecir, del mismo modo que lo podría hacer un sistema experto, la estructura química de un compuesto todavía no desarrollado) o indicar las soluciones en el caso de ciertos problemas que todavía nadie ha logrado resolver.

Limitaciones

A no ser que estén específicamente programados para avisar a la persona que está operando con ellos, los sistemas expertos pueden resultar bastante estúpidos cuando se encuentran con algo que no encaja en el repertorio que tienen preprogramado. Les sucede lo mismo que a esas personas que son muy brillantes en un campo determinado, el ajedrez, por ejemplo, pero que después son incapaces de llegar a dominar algo tan sencillo como anudarse correctamente la corbata. El estatus del sabio-idiota es una característica de muchos sistemas expertos que se basan solamente en la interpretación de reglas del tipo IF/THEN, como las que hemos comentado hace un momento.

Tales sistemas no cuentan con la capacidad de ampliar su base de datos mientras están operando y sólo pueden pensar en línea recta procediendo del punto A al B y luego del B al C y así sucesivamente. Este tipo de sistema no tiene modo de saber cuándo es inapropiado el conocimiento que tan laboriosamente le ha sido programado; no reconocen la excepción a la regla.

Los sistemas que nosotros desarrollaremos aquí entran en la descripción del sabio-

idiota. Pero, a pesar de esta limitación, que, por otro lado, puede aplicarse a la mayoría de los sistemas expertos que actualmente funcionan por todo el mundo, el lector encontrará que los sistemas que desarrolle son artefactos fascinantes. El último sistema que veremos en esta sección tiene la capacidad de aprender. En realidad, sólo con indicarle —cuando está tratando de distinguir entre las diferentes cosas que le hemos programado— si su hipótesis es válida o falsa, terminará aprendiendo por sí solo a distinguir entre los diferentes objetos, sin que en ningún momento le hayamos dicho explícitamente cómo ha de distinguirlos.

Estructura química y DENDRAL

Antes de meternos con nuestros propios programas expertos, echaremos un vistazo a algunos de los sistemas que se están usando en este momento y veremos lo que podemos sacar de su examen.

El primer programa que comentaremos, y posiblemente el primer sistema experto de verdad que se haya usado en el mundo, se llama DENDRAL. Los trabajos para la realización de este sistema —que es capaz de calcular o descubrir hechos relativos a las estructuras moleculares a partir de unos datos químicos sin elaborar— empezaron en la Universidad de Stanford en 1965. Uniendo los conocimientos y experiencia de varias disciplinas, los creadores de DENDRAL consiguieron con el tiempo producir un sistema que ahora desempeña su función mejor que nadie en el mundo (incluyendo a las personas que lo construyeron). DENDRAL se utiliza a lo largo y ancho del mundo.

Stanford fue también la cuna de MYCIN, un sistema que diagnostica infecciones en la sangre y meningitis, y además sugiere el tratamiento que se debe seguir en cada caso. MYCIN basa sus conclusiones en los datos físicos que introduce el médico y puede explicar, si se le pide, el camino seguido para llegar a ese diagnóstico. El sistema contiene unas 450 reglas.

La base de datos de MYCIN es tan valiosa que posteriormente se desarrolló otro programa —GUIDON— para que acompañara a éste y posibilitara el que el ordenador pudiera hacer de profesor, actuando así de puente entre un humano experto (o un equipo de ellos, como en este caso) y otro que está preparándose para serlo.

Y aquí no acaba el valor de MYCIN. Gran parte del programa consiste en las diferentes maneras de *manipular* las reglas que se le han dado y *sacar conclusiones* a partir de ellas. Los mecanismos de manipulación y deducción son, hasta cierto punto, independientes de la base de datos. Esto quiere decir que se puede sacar la información relativa a las infecciones de la sangre para introducir otra nueva. Esto ya se ha llevado a cabo y ahora el sistema experto PUFF dispensa una asistencia similar a la procurada por MYCIN, pero en este caso en relación con las enfermedades de pulmón.

Tan efectivo fue este proceso (en un experimento realizado con 150 pacientes, PUFF dio los mismos diagnósticos que los especialistas) que se desarrolló otra versión de MYCIN, llamada simplemente EMYCIN (de *"Empty"*, "Vacío" MYCIN), en la que se pueden introducir otras bases de datos.

El sistema experto MOLGEN (de *MOLecular GENetics*, "Genética Molecular") ayuda a los biólogos que trabajan en el campo del DNA y la ingeniería genética. Su uso está muy extendido.

Lo más interesante —en términos de examinar las direcciones que está tomando la investigación de la inteligencia artificial— es que los sistemas expertos funcionan, de hecho, extremadamente bien y desde un punto de vista económico su uso resulta totalmente rentable. Esto no sólo garantiza que se están utilizando, sino que además se están construyendo sistemas nuevos. La línea de la "investigación pura", como es natural, también produce resultados, pero éstos suelen aparecer antes, cuando el sistema que se está desarrollando tiene una utilización práctica inmediata y los que lo desarrollan disponen de medios a su alcance (dólares contantes y sonantes).

Imaginemos un sistema que nos aconsejara sobre los lugares más adecuados para realizar prospecciones petrolíferas. Con un solo pozo que se encontrara, el coste de desarrollar el sistema, aun cuando hubiera sido de millones y millones, podría amortizarse relativamente rápido, incluso en cuestión de días.

Feigenbaum y McCorduck (en su libro *The Fifth Generation...,* mencionado antes, páginas 72 y 73) nos dan un ejemplo gráfico del poder de "amortización" de los grandes sistemas expertos. Citan el caso de una importante compañía americana que compró recientemente un sistema diseñado para diagnosticar fallos en un tipo determinado de plantas generadoras de energía eléctrica. Al probar una versión del programa, todavía sin completar, utilizando los datos reales que condujeron a que una de las plantas de la compañía tuviera que ser cerrada en 1981, se encontró que el sistema dio, en tan sólo unos segundos, con la causa del problema que había conducido al cierre. A los expertos humanos que trabajaban en la planta les habría llevado días sin fin el llegar a la misma conclusión. Entretanto, la planta había estado cerrada cuatro días, lo que le costó a la compañía aproximadamente un millón doscientos mil dólares.

Hay otros muchos sistemas funcionando, o en vías de hacerlo, por el mundo. Entre éstos podemos citar:

- PROGRAMMER'S APPRENTICE: Como su propio nombre indica, se trata de un sistema que ayuda en la escritura de programas.
- EURISKO: Un sistema experto, capaz de aprender a medida que funciona, que crea circuitos microeléctricos tridimensionales.
- TAXMAN: Este sistema, que está siendo desarrollado en la Universidad Rutgers, examinará las cambiantes normativas fiscales y aconsejará posteriormente a las compañías sobre cuál es la mejor manera de operar dentro de esas normativas.
- GENESIS: Este suena de lo más emocionante. Se encuentra ya en el mercado y permite a los científicos planificar y simular experimentos en el campo de la unión de genes.

Me temo que por el momento no vamos a ponernos a sacar genes, pero seguro que encontraremos aplicaciones interesantes para nuestros sistemas expertos (como la diferenciación entre un hombre, un caballo y un gorrión). Echemos ahora un vistazo al primero de nuestros sistemas.

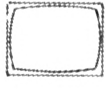


12
Pequeño gran esfuerzo

Nuestro primer sistema experto es GRAN ESFUERZO. Este programa tiene la capacidad de decir, sin error, las diferencias entre tres seres vivos: un hombre, un caballo y un gorrión. Aunque se trata de una situación bastante tonta, puede enseñarnos mucho sobre cómo se desarrollan ciertos tipos de sistemas expertos.

Imaginemos un sistema experto de "diagnóstico médico". Le llamaremos MEDICO. MEDICO y GRAN ESFUERZO son primos hermanos, como se verá enseguida, y el estudio de este último nos sirve de base para saber cómo funcionan MEDICO y otros sistemas expertos más extensos.

Estamos a punto de entrar en la consulta de MEDICO. El sistema nos hace un gran número de preguntas a las que nosotros respondemos SI o NO, como sigue:



```
                ESTA USTED ENFERMO?
               TIENE MAS DE 40 ANOS?
                       FUMA?
   SE HA HECHO UN CHEQUEO EN LOS ULTIMOS 12 MESES?
         TIENE PREOCUPACIONES FRECUENTEMENTE?
 SE DESCRIBIRIA A SI MISMO COMO UNA PERSONA NERVIOSA?
```

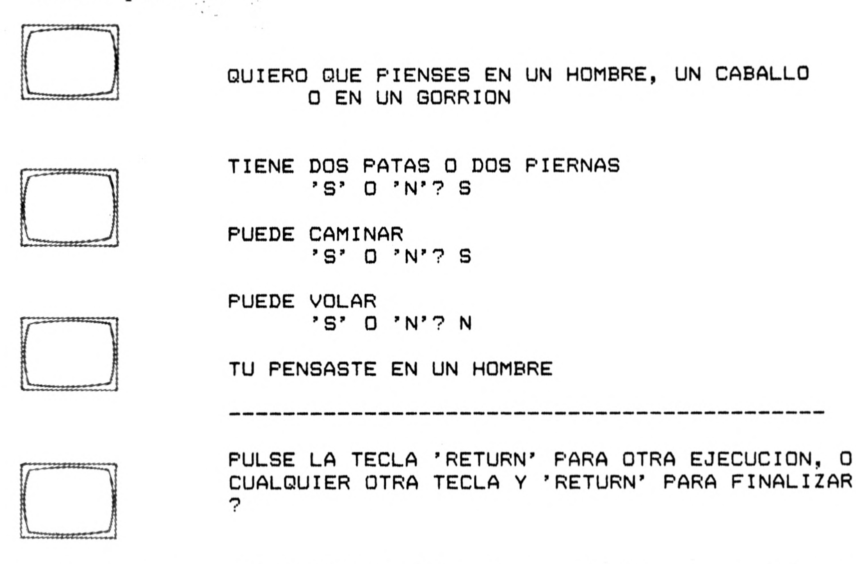
Y así sucesivamente. Tras una serie de estas preguntas MEDICO se detiene un nanosegundo o dos y después imprime en la pantalla el siguiente mensaje:

```
GRACIAS. TU ESPERANZA DE VIDA ES DE 79 ANOS, LO QUE
SUPERA AL 11% DE LA POBLACION. PARA AUMENTAR SUS
POSIBILIDADES DE ALCANZAR, O SUPERAR ESTO, LE SUGIERO
QUE:
    - INTENTE DEJAR DE FUMAR
    - SE HAGA CHEQUEOS MEDICOS REGULARMENTE
    - AUMENTE LAS HORAS SEMANALES DE EJERCICIO

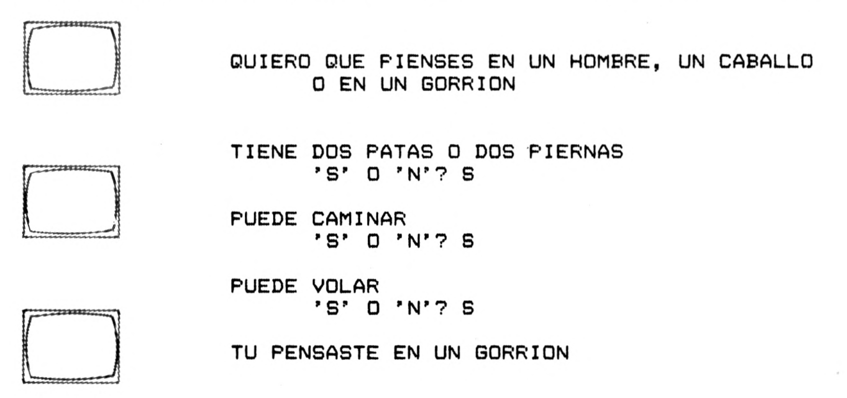
   GRACIAS POR CONSULTAR AL MEDICO
```

¿Qué es lo que ha hecho MEDICO? ¿Cómo ha convertido nuestras escuetas respuestas SI/NO en una predicción de esperanza de vida? En realidad, estoy convencido de que el lector ya ha decidido que éste no es un programa muy complicado y que, por tanto, no requerirá un gran nivel de conocimientos. Sin embargo, muestra cómo podría empezar un verdadero programa de diagnóstico médico, si el sistema experto estuviera actuando recíprocamente con un paciente, más que con un médico, como es generalmente el caso.

Encantados con la idea de que vamos a vivir más que el 11 por 100 de la población, nos disponemos a conocer a otro experto, el joven GRAN ESFUERZO. Esto es lo que se ve en la pantalla:



Por supuesto, GRAN ESFUERZO tiene razón. No era especialmente difícil adivinar por nuestras respuestas que estábamos pensando en un hombre. Totalmente impresionados con este resultado, vamos y pulsamos la tecla RETURN y volvemos a probar:



Ahora estamos seguros de que no va a ser capaz de repetirlo y lo intentamos con la tercera posibilidad:

```
QUIERO QUE PIENSES EN UN HOMBRE, UN CABALLO
      O EN UN GORRION


TIENE DOS PATAS O DOS PIERNAS
      'S' O 'N'? N

PUEDE CAMINAR
      'S' O 'N'? S

PUEDE VOLAR
      'S' O 'N'? N

TU PENSASTE EN UN CABALLO
```

Esta vez decidimos dejarlo por imposible. ¿Cómo recoge las respuestas a nuestras preguntas de modo que puede determinar que si dijimos que tenía dos piernas y podía caminar, pero no podía volar, la criatura en la que estábamos pensando era un hombre? ¿Cómo registraba MEDICO nuestras respuestas para podernos decir que viviremos hasta los 79 años?

Es muy simple, al menos en el caso de GRAN ESFUERZO (y MEDICO funciona de modo parecido, sólo que con un poco más de refinamiento). GRAN ESFUERZO contó cada vez que respondimos 'S' a una pregunta. Si sólo respondimos una vez con 'S', teníamos que estar pensando en un caballo (ya que la pregunta relativa al caminar era la única a la que podíamos responder con un 'S', en el caso de que estuviéramos pensando en un caballo). Dos respuestas 'S', y en lo que estábamos pensando era en un hombre. Tres, y GRAN ESFUERZO, sin ningún esfuerzo, adivinó que pensábamos en un gorrión.

MEDICO no sólo contaba las respuestas, sino que también anotaba a qué pregunta se referían. Una respuesta 'S' a la pregunta ¿FUMA? quitaba tres años a nuestra esperanza de vida, mientras que una respuesta 'S' a la pregunta ¿SE HA HECHO UN CHEQUEO EN LOS ULTIMOS DOCE MESES? añadía cinco años de duración a nuestra vida.

El listado de GRAN ESFUERZO empieza así:

```
10 REM GRAN ESFUERZO
20 CLS
30 PRINT "QUIERO QUE PIENSES EN UN HOMBRE, UN
         CABALLO"
40 PRINT TAB(6); "O EN UN GORRION"
50 FOR J = 1 TO 2000: NEXT J
60 PRINT: PRINT
70 GOSUB 170: REM HACE LAS PREGUNTAS
```

Después de fijar la escena, el programa entra en materia intentado determinar en qué criatura estamos pensando:

```
170 REM REALIZA LAS PREGUNTAS
180 CCONTADOR = 0
190 PRINT "TIENE DOS PATAS O DOS PIERNAS"
200 GOSUB 310
210 PRINT "PUEDE CAMINAR"
220 GOSUB 310
230 PRINT "PUEDE VOLAR"
240 GOSUB 310


310 REM SE PROCESA LA RESPUESTA
320 INPUT "       'S' O 'N'"; Z$
330 IF Z$ <> "S" AND Z$ <> "N" THEN 320
340 IF Z$ = "S" THEN CCONTADOR = CCONTADOR + 1
350 PRINT
360 RETURN
```

Observaremos que la variable contador se pone a cero al inicio de la ejecución y que se va aumentando con uno cada vez que se da una respuesta 'S'. Utilizando esta información GRAN ESFUERZO no encuentra dificultad para determinar cuál es la criatura en la que estamos pensando:

```
250 PRINT "TU PENSASTE EN UN ";
260 IF CCONTADOR = 1 THEN PRINT "CABALLO"
270 IF CCONTADOR = 2 THEN PRINT "HOMBRE"
280 IF CCONTADOR = 3 THEN PRINT "GORRION"
290 RETURN
```

Como vemos, es un programa muy sencillo, pero, pese a ello, pone los cimientos necesarios para poder construir sistemas expertos. Este es el listado completo (para usuarios del ZX Spectrum, véase Apéndice):

```
10 REM GRAN ESFUERZO
20 CLS
30 PRINT "QUIERO QUE PIENSES EN UN HOMBRE, UN
          CABALLO"
40 PRINT TAB(6); "O EN UN GORRION"
50 FOR J = 1 TO 2000: NEXT J
60 PRINT: PRINT
70 GOSUB 170: REM HACE LAS PREGUNTAS
80 PRINT
90 PRINT "-------------------------------------
          ---------"
100 PRINT: PRINT "PULSE LA TECLA 'RETURN' PARA
          OTRA EJECUCION, O"
```

```
110 PRINT "CUALQUIER OTRA TECLA Y 'RETURN' PARA
         FINALIZAR"
120 INPUT Q$
130 IF Q$ <> "" THEN END
140 CLS
150 GOTO 30
160 REM *********************
170 REM REALIZA LAS PREGUNTAS
180 CCONTADOR = 0
190 PRINT "TIENE DOS PATAS O DOS PIERNAS"
200 GOSUB 310
210 PRINT "PUEDE CAMINAR"
220 GOSUB 310
230 PRINT "PUEDE VOLAR"
240 GOSUB 310
250 PRINT "TU PENSASTE EN UN ";
260 IF CCONTADOR = 1 THEN PRINT "CABALLO"
270 IF CCONTADOR = 2 THEN PRINT "HOMBRE"
280 IF CCONTADOR = 3 THEN PRINT "GORRION"
290 RETURN
300 REM ***********************
310 REM SE PROCESA LA RESPUESTA
320 INPUT "       'S' O 'N'"; Z$
330 IF Z$ <> "S" AND Z$ <> "N" THEN 320
340 IF Z$ = "S" THEN CCONTADOR = CCONTADOR + 1
350 PRINT
360 RETURN
```

Pequeño GRAN ESFUERZO-X

GRAN ESFUERZO-X es el hermano mayor de GRAN ESFUERZO. Aunque este nuevo programa tiene lazos de parentesco definidos con el que acabamos de ver, es considerablemente más complicado.

Observando una sola ejecución de éste, puede verse su grado de complejidad en comparación con el anterior. En primer lugar, haremos que éste actúe de modo parecido a como lo hacía el otro. Sin embargo, ya desde la primera impresión en la pantalla, puede decirse que se trata de dos programas diferentes. Este es, en gran medida, un programa autosuficiente, es decir, la capacidad de decisión no descansa sólo en la base de datos como en el caso de GRAN ESFUERZO, sino en algoritmos de decisión más complejos.

```
NOMBRE DEL SISTEMA? CRIATURAS

NUMERO DE ELEMENTOS? 3

NUMERO DE FACTORES QUE SE VAN A CONSIDERAR? 3
```

Le decimos al programa el tema que va a tratar (CRIATURAS, en este caso) y el número de ELEMENTOS (es decir, resultados) y el número de FACTORES QUE SE VAN

A CONSIDERAR. Estas son las variables (como PUEDE VOLAR) que han de considerarse. Habiéndole proporcionado una primera estructura, GRAN ESFUERZO-X nos pide entonces que completemos los contornos:

```
                    CRIATURAS


QUE ES EL ELEMENTO 1 ? UN HOMBRE

QUE ES EL ELEMENTO 2 ? UN CABALLO

QUE ES EL ELEMENTO 3 ? UN GORRION
```

Una vez que le hemos dicho los elementos, nos pide que introduzcamos las preguntas relacionadas con los factores que determinan el elemento que estamos buscando:

```
POR FAVOR HAGA LA PREGUNTA 1
? PUEDE VOLAR POR SI MISMO

POR FAVOR HAGA LA PREGUNTA 2
? TIENE DOS PATAS O PIERNAS

POR FAVOR HAGA LA PREGUNTA 3
? PUEDE CAMINAR
```

En principio, esto puede dar la impresión de que nos estamos creando problemas inútiles sólo para imitar a GRAN ESFUERZO, pero, como veremos enseguida, merece la pena hacerlo. Este sencillo ejercicio nos muestra cómo podemos preparar a GRAN ESFUERZO-X para convertirlo en un experto sobre lo que queramos.

GRAN ESFUERZO-X recorre entonces cada uno de los elementos y dice: "Si te hiciera la siguiente pregunta con respecto a este elemento contestarías 'sí' o 'no'. A partir de esta información, GRAN ESFUERZO-X reúne una base de datos equivalente a la que telegrafiamos en GRAN ESFUERZO. Por supuesto, GRAN ESFUERZO-X podría construir su base de datos sobre cualquier tema:

```
POR FAVOR RESPONDA LA SIGUIENTE PREGUNTA
PARA EL ELEMENTO >  UN HOMBRE   <

ESCRIBA 'S' PARA UNA RESPUESTA AFIRMATIVA
                O 'N' PARA UNA NEGATIVA

    > PUEDE VOLAR POR SI MISMO? N

    > TIENE DOS PATAS O PIERNAS? S

    > PUEDE CAMINAR? S
```

```
POR FAVOR RESPONDA LA SIGUIENTE PREGUNTA
PARA EL ELEMENTO >  UN CABALLO   <


ESCRIBA 'S' PARA UNA RESPUESTA AFIRMATIVA
                  O 'N' PARA UNA NEGATIVA


    > PUEDE VOLAR POR SI MISMO? N

    > TIENE DOS PATAS O PIERNAS? N

    > PUEDE CAMINAR? S
```

```
POR FAVOR RESPONDA LA SIGUIENTE PREGUNTA
PARA EL ELEMENTO >  UN GORRION   <


ESCRIBA 'S' PARA UNA RESPUESTA AFIRMATIVA
                  O 'N' PARA UNA NEGATIVA


    > PUEDE VOLAR POR SI MISMO? S

    > TIENE DOS PATAS O PIERNAS? S

    > PUEDE CAMINAR? S
```

Una vez que hemos recorrido cada uno de los posibles elementos y le hemos dicho al programa cuáles serían nuestras respuestas a esas preguntas, GRAN ESFUERZO-X crea una base de datos que, en este caso, es poco más que sumar el total de respuestas 'S'. GRAN ESFUERZO-X nos cuenta sus descubrimientos:

```
ESTA ES MI BASE DE DATOS:

UN HOMBRE --- 6

UN CABALLO --- 4

UN GORRION --- 7
```

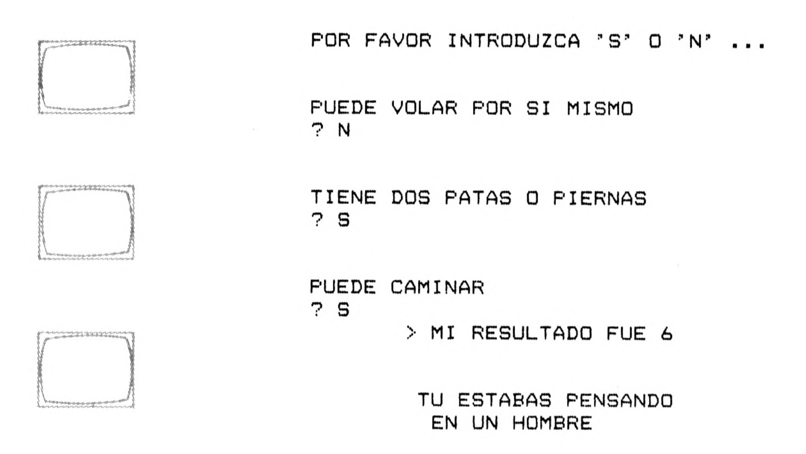
¿Pero de dónde ha sacado esos números? Nosotros no hemos podido responder cuatro veces 'S' en el caso del caballo, ni siete en el del gorrión, porque sólo ha habido tres preguntas. GRAN ESFUERZO-X no añade sólo un uno por cada 'S', sino que da un número variable para cada respuesta. Si sólo añadiera uno por cada respuesta 'S' y nosotros respondiéramos 'S' a, pongamos por caso, las preguntas una y tres por una cosa y a las preguntas dos y tres por otra, los dos objetos tendrían el mismo total.

Para soslayar este problema, para garantizar que el orden real en el que se dan las respuestas 'S' es importante, procederemos del siguiente modo:

Una respuesta 'S' a la pregunta 1 vale 1
Una respuesta 'S' a la pregunta 2 vale 2
Una respuesta 'S' a la pregunta 3 vale 4
Una respuesta 'S' a la pregunta 4 vale 8
... 5 ... 16
... 6 ... 32
... 7 ... 64
... y así sucesivamente...

Esto garantiza que, aun cuando se dé el mismo número de respuestas 'S' a dos cosas diferentes, nuestro experto, a la hora de emitir un juicio, tendrá un número de identificación diferente.

¿Funciona? Pues claro que funciona, y éste es GRAN ESFUERZO-X mostrándonos cómo trabaja:



Dije antes que GRAN ESFUERZO-X era capaz de mucho más que GRAN ESFUERZO y ahora voy a demostrar que esta afirmación es cierta. Vamos a instruir a nuestro sistema experto en otro campo; un campo en el que yo soy totalmente lego. La base de datos introducida en GRAN ESFUERZO-X proviene de un libro escrito por un verdadero experto en ese campo, llamado Oliver Chambers (el libro es *The Observer's Book of Rocks and Minerals,* Nueva York, Frederick Warne, 1979). Con la ayuda del señor Chambers, GRAN ESFUERZO-X se hará con la técnica necesaria para identificar cinco tipos diferentes de minerales utilizando para ello cuatro factores.

Esto es algo que yo no podría hacer sin la ayuda de GRAN ESFUERZO-X. Lo mismo puede aplicarse a la mayoría de la gente que hoy usa sistemas expertos. Un sistema experto codifica, como si dijéramos, la pericia y el conocimiento de un experto ausente, de modo que los no expertos puedan utilizar esa base de datos a voluntad.

Pasemos ahora un poco del conocimiento del señor Chambers a nuestro sistema (em-

pezando por el principio, de modo que los minerales no se confundan con los caballos y los gorriones):

Le decimos el tema que va a tratar:

```
NOMBRE DEL SISTEMA? IDENTIFICACION DE MINERALES

NUMERO DE ELEMENTOS? 5

NUMERO DE FACTORES QUE SE VAN A CONSIDERAR? 4
```

Después le decimos los cinco minerales que va a intentar identificar:

```
QUE ES EL ELEMENTO 1 ? PLEONASTO

QUE ES EL ELEMENTO 2 ? LIMONITA

QUE ES EL ELEMENTO 3 ? YODORITA

QUE ES EL ELEMENTO 4 ? IRIDOSMIO

QUE ES EL ELEMENTO 5 ? SILVANITA
```

Seguidamente, GRAN ESFUERZO-X aprende unas preguntas que le ayudarán a la hora de diferenciar los minerales:

```
POR FAVOR HAGA LA PREGUNTA 1
? ES DURO

POR FAVOR HAGA LA PREGUNTA 2
? TIENE RAYAS DE UN COLOR DISTINTO
DEL COLOR PRINCIPAL

POR FAVOR HAGA LA PREGUNTA 3
? ES SU PESO ESPECIFICO APROXIMADAMENTE 5

POR FAVOR HAGA LA PREGUNTA 4
? ES FUNDIBLE
```

Ahora le toca al usuario el largo proceso de codificar los conocimientos y pericia:

```
POR FAVOR RESPONDA LA SIGUIENTE PREGUNTA
PARA EL ELEMENTO >  PLEONASTO   <

ESCRIBA 'S' PARA UNA RESPUESTA AFIRMATIVA
                  O 'N' PARA UNA NEGATIVA

    > ES DURO? S

    > TIENE RAYAS DE UN COLOR DISTINTO
      DEL COLOR PRINCIPAL? N

    > ES SU PESO ESPECIFICO APROXIMADAMENTE 5? N

    > ES FUNDIBLE? N
```

Va haciendo lo mismo con el resto de los minerales. Finalmente el sistema nos cuenta sus hallazgos:

```
ESTA ES MI BASE DE DATOS:

PLEONASTO --- 1

LIMONITA --- 8

YODORITA --- 12

IRIDOSMIO --- 7

SILVANITA --- 13
```

Pongámoslo ahora en funcionamiento y veamos lo bien que marcha:

```
POR FAVOR PIENSE EN UNA DE LAS SIGUIENTES ELEMENTOS:
    PLEONASTO
     LIMONITA
      YODORITA
       IRIDOSMIO
        O SILVANITA

POR FAVOR INTRODUZCA 'S' O 'N' ...

 ES DURO
 ? N
```

```
TIENE RAYAS DE UN COLOR DISTINTO
DEL COLOR PRINCIPAL
? N


ES SU PESO ESPECIFICO APROXIMADAMENTE 5
? N


ES FUNDIBLE
? S
         > MI RESULTADO FUE 8


          TU ESTABAS PENSANDO
          EN LIMONITA
```

En cuestión de minutos, GRAN ESFUERZO-X ha adquirido un conocimiento que a mí me permite, como total lego en esa materia, hacer uso del juicio de un experto en una situación práctica.

Estudiaremos ahora la construcción del programa y veremos la versión "inteligente" de GRAN ESFUERZO que hemos creado.

El programa está controlado por un bucle de llamadas a diferentes subrutinas:

```
 10 REM GRAN ESFUERZO - X
 20 GOSUB 940: REM INICIALIZACION
 30 GOSUB 450: REM 'CONSIGUE EXPERIENCIA'
 40 GOSUB 120: REM DEMUESTRA EXPERIENCIA
 50 GOSUB 1060
 60 PRINT "PULSE LA TECLA 'RETURN' PARA EJECUTAR
          EL PROGRAMA DE NUEVO, O"
 70 PRINT "CUALQUIER OTRA TECLA Y LUEGO 'RETURN'
          PARA FINALIZAR"
 80 INPUT Q$
 90 IF Q$ = "" THEN 40
100 END
```

En la rutina de iniciación, GRAN ESFUERZO-X da un nombre al sistema (que puede resultar útil si se quiere guardar un archivo), así como un número a los elementos y a los factores. Se dimensionan unas matrices que contendrán los nombres y los totales de los elementos, así como las preguntas relacionadas con los factores.

```
940 REM INICIALIZACION
950 CLS
960 INPUT "NOMBRE DEL SISTEMA"; N$
970 GOSUB 1060
980 INPUT "NUMERO DE ELEMENTOS"; ELEMENTOS
990 GOSUB 1060
```

```
1000 INPUT "NUMERO DE FACTORES QUE SE VA A CONSI
           DERAR"; FACTORES
1010 DIM A$(ELEMENTOS), B$(FACTORES)
1020 DIM D(ELEMENTOS)
1030 CLS
1040 RETURN
```

La siguiente sección del código que visita el programa consigue los nombres de los elementos:

```
450 REM SE LLENAN LAS MATRICES
460 PRINT TAB(20 - LEN(N$)/2); N$
470 GOSUB 1060
480 REM ** CONSIGUE LOS NOMBRES DE LOS ELEMENTOS
        **
490 FOR J = 1 TO ELEMENTOS
500 GOSUB 1060
510 PRINT "QUE ES EL ELEMENTO"; J;
520 INPUT A$(J)
530 NEXT J
```

Y después GRAN ESFUERZO-X pregunta por los factores:

```
550 REM ** ALMACENA LAS PREGUNTAS QUE SE VAN A
        HACER **
560 FOR J = 1 TO FACTORES
570 GOSUB 1060
580 PRINT "POR FAVOR HAGA LA PREGUNTA"; J
590 INPUT B$(J)
600 NEXT J
```

Todo esto no es más que la preparación. Ahora GRAN ESFUERZO-X quiere conseguir más información, de modo que recorre los elementos (utilizando el bucle J, que empieza en la línea 630 y acaba en la 810) y, dentro de ellos, los factores (con el bucle K, entre las líneas 720 y 800):

```
620 REM ** ADQUIERE EXPERIENCIA **
630 FOR J = 1 TO ELEMENTOS
640 CLS
650 GOSUB 1060
660 PRINT "POR FAVOR RESPONDA LA SIGUIENTE PREGUN
          TA"
```

```
670 PRINT "PARA EL ELEMENTO >   "; A$(J); "    <"
680 GOSUB 1060
690 PRINT "ESCRIBA 'S' PARA UNA RESPUESTA AFIRMA
           TIVA"
700 PRINT "                    O 'N' PARA UNA NEGA
           TIVA"
710 X = .5
720 FOR K = 1 TO FACTORES
730 X = X + X
740 GOSUB 1060
750 PRINT TAB(4); "> "; B$(K);
760 MULTI = 0
770 INPUT Y$
780 IF Y$ <> "N" THEN MULTI = 1
790 D(J) = D(J) + X * MULTI: REM
800 NEXT K
810 NEXT J
```

Una vez hecho esto, GRAN ESFUERZO-X nos enseña lo que ha aprendido:

```
830 PRINT "ESTA ES MI BASE DE DATOS:"
840 FOR J = 1 TO ELEMENTOS
850 GOSUB 1060
860 PRINT A$(J); " ---"; D(J)
870 NEXT J
880 GOSUB 1060
890 PRINT TAB(8); "PULSE LA TECLA 'RETURN'"
900 INPUT Q$
910 CLS
920 RETURN
```

Guardándose esta información bien guardada en el bolsillo, GRAN ESFUERZO-X está ahora dispuesto a actuar, del mismo modo que hizo GRAN ESFUERZO, preguntando, sumando números y tomando una decisión a partir del total:

```
120 REM DEMUESTRA EXPERIENCIA
130 CLS
140 GOSUB 1060
150 PRINT "POR FAVOR PIENSE EN UNO DE LOS SIGUIEN
           TES ELEMENTOS:"
160 FOR J = 1 TO ELEMENTOS
170 PRINT TAB(J + 2);
180 IF J = ELEMENTOS THEN PRINT "O ";
190 PRINT A$(J)
200 NEXT J
210 GOSUB 1060
220 RESULTADO = 0
230 X = .5
```

```
240 PRINT "POR FAVOR INTRODUZCA 'S' O 'N' ..."
250 FOR J = 1 TO FACTORES
260 X = X + X
270 GOSUB 1060
280 PRINT B$(J)
290 INPUT E$
300 IF E$ <> "N" THEN RESULTADO = RESULTADO + X
310 NEXT J
320 PRINT TAB(8); "> MI RESULTADO FUE"; RESULTADO
330 GOSUB 1060
340 M = 0
350 M = M + 1
360 IF D(M) = RESULTADO THEN 400
370 IF M < ELEMENTOS THEN 350
380 PRINT TAB(8); "> NO PUEDO IDENTIFICARLO"
390 RETURN
400 PRINT TAB(8); " TU ESTABAS PENSANDO"
410 PRINT TAB (10); "EN "; A$(M)
420 GOTO 390
430 RETURN
```

Como vemos, GRAN ESFUERZO-X se permite un poco de falibilidad, con una sentencia como NO PUEDO IDENTIFICARLO, si el total al que ha llegado no concuerda con ninguna de las cosas que le hemos programado (línea 380).

Asimismo, vemos que el sistema nos va mostrando sus cálculos después de cada ejecución; de este modo podemos seguir fácilmente lo que hace. Si el lector quiere impresionar a la gente con su sistema experto, lo mejor es que procure que el ordenador no haga sus cálculos en público, con lo que la impresión que ya de por sí produce el sistema será todavía mayor.

Este es el listado completo de GRAN ESFUERZO-X (para usuarios del ZX Spectrum, véase Apéndice):

```
 10 REM GRAN ESFUERZO - X
 20 GOSUB 940: REM INICIALIZACION
 30 GOSUB 450: REM 'CONSIGUE EXPERIENCIA'
 40 GOSUB 120: REM DEMUESTRA EXPERIENCIA
 50 GOSUB 1060
 60 PRINT "PULSE LA TECLA 'RETURN' PARA EJECUTAR
          EL PROGRAMA DE NUEVO, O"
 70 PRINT "CUALQUIER OTRA TECLA Y LUEGO 'RETURN'
          PARA FINALIZAR"
 80 INPUT Q$
 90 IF Q$ = "" THEN 40
100 END
110 REM *********************
120 REM DEMUESTRA EXPERIENCIA
130 CLS
140 GOSUB 1060
150 PRINT "POR FAVOR PIENSE EN UNO DE LOS SIGUIEN
          TES ELEMENTOS:"
160 FOR J = 1 TO ELEMENTOS
170 PRINT TAB(J + 2);
```

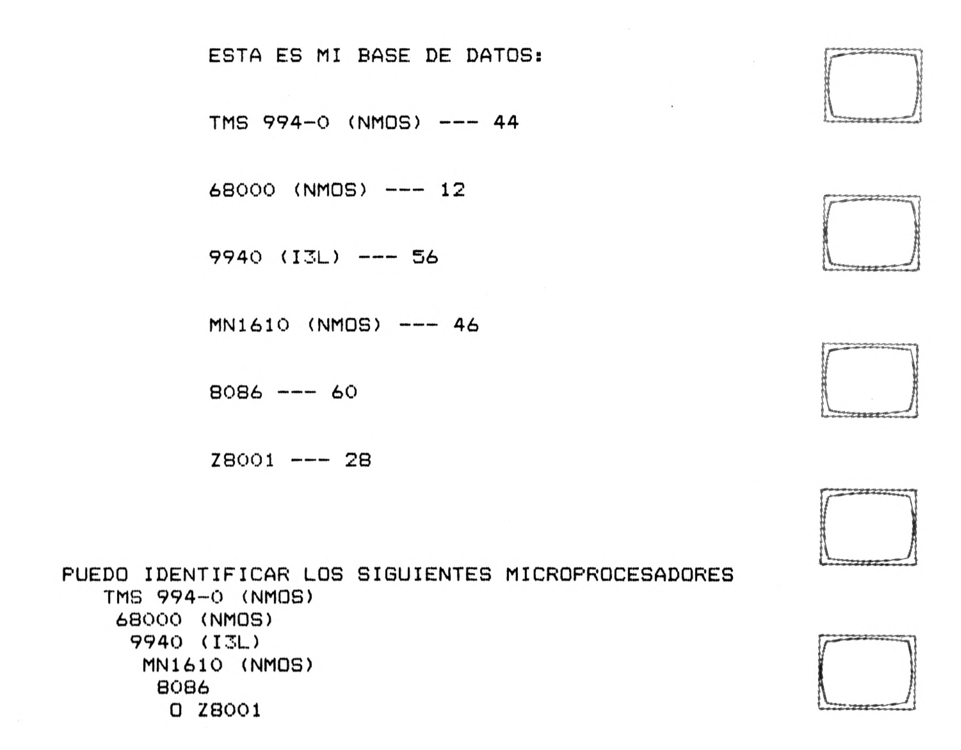
```
180 IF J = ELEMENTOS THEN PRINT "O ";
190 PRINT A$(J)
200 NEXT J
210 GOSUB 1060
220 RESULTADO = 0
230 X = .5
240 PRINT "POR FAVOR INTRODUZCA 'S' O 'N' ..."
250 FOR J = 1 TO FACTORES
260 X = X + X
270 GOSUB 1060
280 PRINT B$(J)
290 INPUT E$
300 IF E$ <> "N" THEN RESULTADO = RESULTADO + X
310 NEXT J
320 PRINT TAB(8); "> MI RESULTADO FUE"; RESULTADO
330 GOSUB 1060
340 M = 0
350 M = M + 1
360 IF D(M) = RESULTADO THEN 400
370 IF M < ELEMENTOS THEN 350
380 PRINT TAB(8); "> NO PUEDO IDENTIFICARLO"
390 RETURN
400 PRINT TAB(8); " TU ESTABAS PENSANDO"
410 PRINT TAB (10); "EN "; A$(M)
420 GOTO 390
430 RETURN
440 REM *********************
450 REM SE LLENAN LAS MATRICES
460 PRINT TAB(20 - LEN(N$)/2); N$
470 GOSUB 1060
480 REM ** CONSIGUE LOS NOMBRES DE LOS ELEMENTOS
        **
490 FOR J = 1 TO ELEMENTOS
500 GOSUB 1060
510 PRINT "QUE ES EL ELEMENTO"; J;
520 INPUT A$(J)
530 NEXT J
540 CLS
550 REM ** ALMACENA LAS PREGUNTAS QUE SE VAN A
        HACER **
560 FOR J = 1 TO FACTORES
570 GOSUB 1060
580 PRINT "POR FAVOR HAGA LA PREGUNTA"; J
590 INPUT B$(J)
600 NEXT J
610 CLS
620 REM ** ADQUIERE EXPERIENCIA **
630 FOR J = 1 TO ELEMENTOS
640 CLS
650 GOSUB 1060
660 PRINT "POR FAVOR RESPONDA LA SIGUIENTE PREGUN
           TA"
670 PRINT "PARA EL ELEMENTO >  "; A$(J); "   <"
680 GOSUB 1060
690 PRINT "ESCRIBA 'S' PARA UNA RESPUESTA AFIRMA
           TIVA"
```

```
700 PRINT "                          O 'N' PARA UNA NEGA
          TIVA"
710 X = .5
720 FOR K = 1 TO FACTORES
730 X = X + X
740 GOSUB 1060
750 PRINT TAB(4); "> "; B$(K);
760 MULTI = 0
770 INPUT Y$
780 IF Y$ <> "N" THEN MULTI = 1
790 D(J) = D(J) + X * MULTI: REM
800 NEXT K
810 NEXT J
820 CLS
830 PRINT "ESTA ES MI BASE DE DATOS:"
840 FOR J = 1 TO ELEMENTOS
850 GOSUB 1060
860 PRINT A$(J); " ---"; D(J)
870 NEXT J
880 GOSUB 1060
890 PRINT TAB(8); "PULSE LA TECLA 'RETURN'"
900 INPUT Q$
910 CLS
920 RETURN
930 REM **************
940 REM INICIALIZACION
950 CLS
960 INPUT "NOMBRE DEL SISTEMA"; N$
970 GOSUB 1060
980 INPUT "NUMERO DE ELEMENTOS"; ELEMENTOS
990 GOSUB 1060
1000 INPUT "NUMERO DE FACTORES QUE SE VA A CONSI
          DERAR"; FACTORES
1010 DIM A$(ELEMENTOS), B$(FACTORES)
1020 DIM D(ELEMENTOS)
1030 CLS
1040 RETURN
1050 REM ********
1060 PRINT: PRINT
1070 RETURN
```

La elección de un microprocesador

Sería una lata que tuviéramos que educar a nuestro experto cada vez que quisiéramos hacer uso de sus conocimientos. Es muy poco probable, en una situación real, que se necesite un sistema experto totalmente *soft*. El siguiente programa muestra un cuerpo de conocimientos experto —la capacidad para distinguir entre diferentes tipos de microprocesadores (de la variedad del sílice)— codificado en sentencias DATA.

Así funciona el programa:



```
ESTA ES MI BASE DE DATOS:

TMS 994-0 (NMOS) --- 44

68000 (NMOS) --- 12

9940 (I3L) --- 56

MN1610 (NMOS) --- 46

8086 --- 60

Z8001 --- 28


PUEDO IDENTIFICAR LOS SIGUIENTES MICROPROCESADORES
   TMS 994-0 (NMOS)
    68000 (NMOS)
     9940 (I3L)
      MN1610 (NMOS)
       8086
        O Z8001
```

Después de decirnos lo que puede hacer, el programa nos pide que introduzcamos un número de preguntas relativas al microprocesador que estamos tratando de identificar. Luego nos dirá el nombre del microprocesador:

```
POR FAVOR INTRODUZCA 'S' O 'N' ...
ES LA LONGITUD DE PALABRA DE 32 BITS
? N
TIENE 64K DE MEMORIA
? N
ES LA FRECUENCIA DEL OSCILADOR DE 5 MHz O MENOR
? S
EL TIEMPO DE EJECUCION MINIMO ES DE 3 MICROSEGUNDOS O MENOS
? S
TIENE EL JUEGO DE INSTRUCCIONES MAS DE 71
? N
TIENE EL INTEGRADO 40 PATILLAS
? N
        > MI RESULTADO FUE 12
        >  EL QUE TU TIENES
          ES EL 68000 (NMOS)
```

La base de datos específicos procede de otro experto, Ken Ozanne, cuyos conocimientos ya fueron codificados una vez en forma de libro (*The Interface Computer Enclyclopedia,* Londres, Interface Publications, 1983). Una vez que la información queda encerrada en las sentencias DATA del programa, estará siempre a punto para cuando la queramos utilizar.

Esta es la parte fundamental del programa, aquella en la que se almacena el conocimiento:

```
580 REM INICIALIZACION
590 CLS
600 RESTORE
610 ELEMENTOS = 6
620 FACTORES = 6
630 DIM A$(ELEMENTOS), B$(FACTORES), D(ELEMENTOS)
640 FOR J = 1 TO ELEMENTOS
650 READ A$(J), D(J)
660 NEXT J
670 FOR J = 1 TO FACTORES
680 READ B$(J)
690 NEXT J
700 RETURN
710 REM ************
720 REM PRINT: PRINT
730 RETURN
740 REM ****************************************
        *******
750 REM ** ELEMENTOS (NOMBRES DE MICROPROCESADO
        RES) **
760 DATA "TMS 994-0 (NMOS)", 44, "68000 (NMOS)",
         12
770 DATA "9940 (I3L)", 56, "MN1610 (NMOS)",46
780 DATA "8086", 60, "Z8001", 28
790 REM ** PREGUNTAS **
800 DATA "ES LA LONGITUD DE PALABRA DE 32 BITS"
810 DATA "TIENE 64K DE MEMORIA"
820 DATA "ES LA FRECUENCIA DEL OSCILADOR DE 5 MHz
         O MENOR"
830 DATA "EL TIEMPO DE EJECUCION MINIMO ES DE 3
          MICROSEGUNDOS O MENOS"
840 DATA "TIENE EL JUEGO DE INSTRUCCIONES MAS DE
          71"
850 DATA "TIENE EL INTEGRADO 40 PATILLAS"
```

Aun cuando no se tenga ningún interés en identificar microprocesadores, el sistema experto codificado en este programa puede resultar muy útil. Como vemos, las variables ELEMENTOS y FACTORES se asignan en las líneas 610 y 620 y después éstos se utilizan para dimensionar las matrices en la línea 630. Cambiando las variables de los elementos y factores y modificando las sentencias DATA tendremos nuestro propio sistema experto, siempre a punto para ayudarnos cuando quiera que lo necesitemos. El número crucial que utiliza el sistema para identificar el microprocesador (o para aislar el elemento que se quiera) está contenido en las sentencias DATA, inmediatamente después del nombre del mismo.

Con el fin de calcular los números, yo hice este gráfico. Es muy fácil hacer lo mismo para otros temas:

ELEMENTO	*FACTOR*						*TOTAL*
	1	*2*	*3*	*4*	*5*	*6*	
TMS 9940	*0*	*0*	*1*	*1*	*0*	*1*	*44*
68000	*0*	*0*	*1*	*1*	*0*	*0*	*12*
994013L	*0*	*0*	*0*	*1*	*1*	*1*	*56*
MN1610	*0*	*1*	*1*	*1*	*0*	*1*	*46*
8086	*0*	*0*	*1*	*1*	*1*	*1*	*60*
Z8001	*0*	*0*	*1*	*1*	*1*	*0*	*28*

Este es el programa completo. Con él cada cual puede crear el experto que elija o necesite (para usuarios del ZX Spectrum, véase Apéndice).

```
 10 REM ELECCION DE UN MICROPROCESADOR
 20 GOSUB 580: REM INICIALIZACION
 30 GOSUB 450: REM MUESTRA EL CONTENIDO DE LA
    BASE DE DATOS
 40 GOSUB 120: REM IDENTIFICA EL MICROPROCESADOR
 50 GOSUB 720
 60 PRINT "PULSE LA TECLA 'RETURN' PARA OTRO
           MICROPROCESADOR, O"
 70 PRINT "CUALQUIER OTRA TECLA Y LUEGO 'RETURN',
          PARA FINALIZAR"
 80 INPUT Q$
 90 IF Q$ = "" THEN 40
100 END
110 REM **********************************
120 REM IDENTIFICACION DEL MICROPROCESADOR
130 CLS
140 GOSUB 720
150 PRINT "PUEDO IDENTIFICAR LOS SIGUIENTES MICRO
           PROCESADORES"
160 FOR J = 1 TO ELEMENTOS
170 PRINT TAB(J + 2);
180 IF J = ELEMENTOS THEN PRINT "O ";
190 PRINT A$(J)
200 NEXT J
210 GOSUB 720
220 RESULTADO = 0
230 X = .5
240 PRINT "POR FAVOR INTRODUZCA 'S' O 'N' ..."
250 FOR J = 1 TO FACTORES
260 X = X + X
270 GOSUB 720
280 PRINT B$(J)
290 INPUT E$
300 IF E$ <> "N" THEN RESULTADO = RESULTADO + X
310 NEXT J
```

```
320 PRINT TAB(8); "> MI RESULTADO FUE"; RESULTADO
330 GOSUB 720
340 M = 0
350 M = M + 1
360 IF D(M) = RESULTADO THEN 400
370 IF M < ELEMENTOS THEN 350
380 PRINT TAB(8); "> NO PUEDO IDENTIFICARLO"
390 RETURN
400 PRINT TAB(8); ">  EL QUE TU TIENES"
410 PRINT TAB (10); "ES EL "; A$(M)
420 GOTO 390
430 RETURN
440 REM ***************************************
450 REM MUESTRA EL CONTENIDO DE LA BASE DE DATOS
460 CLS
470 PRINT "ESTA ES MI BASE DE DATOS:": PRINT:
    PRINT
480 FOR J = 1 TO ELEMENTOS
490 GOSUB 720
500 PRINT A$(J); " ---"; D(J):PRINT: PRINT
510 NEXT J
520 GOSUB 720
530 PRINT TAB(8) "PULSE LA TECLA 'RETURN'"
540 INPUT Q$
550 CLS
560 RETURN
570 REM **************
580 REM INICIALIZACION
590 CLS
600 RESTORE
610 ELEMENTOS = 6
620 FACTORES = 6
630 DIM A$(ELEMENTOS), B$(FACTORES), D(ELEMENTOS)
640 FOR J = 1 TO ELEMENTOS
650 READ A$(J), D(J)
660 NEXT J
670 FOR J = 1 TO FACTORES
680 READ B$(J)
690 NEXT J
700 RETURN
710 REM ************
720 REM PRINT: PRINT
730 RETURN
740 REM ***************************************
        *******
750 REM ** ELEMENTOS (NOMBRES DE MICROPROCESADO
        RES) **
760 DATA "TMS 994-0 (NMOS)", 44, "68000 (NMOS)",
         12
770 DATA "9940 (I3L)", 56, "MN1610 (NMOS)",46
780 DATA "8086", 60, "Z8001", 28
790 REM ** PREGUNTAS **
800 DATA "ES LA LONGITUD DE PALABRA DE 32 BITS"
810 DATA "TIENE 64K DE MEMORIA"
820 DATA "ES LA FRECUENCIA DEL OSCILADOR DE 5 MHz
         O MENOR"
```

```
830 DATA "EL TIEMPO DE EJECUCION MINIMO ES DE 3
         MICROSEGUNDOS O MENOS"
840 DATA "TIENE EL JUEGO DE INSTRUCCIONES MAS DE
         71"
850 DATA "TIENE EL INTEGRADO 40 PATILLAS"
```



13
Sistemas que aprenden solos

Recordemos que en el segundo sistema que comentamos en esta sección, GRAN ESFUERZO-X, el programa nos permitía introducir conocimientos sobre cualquier tema. Una vez que los habíamos introducido, el programa ya estaba preparado para ser nuestro experto en el tema que hubiéramos escogido.

Sin embargo, tenía un inconveniente. Exigía que recorriéramos los factores uno por uno en el caso de cada uno de los elementos, con el fin de adquirir la base de conocimientos sobre la que posteriormente trabajaría.

Nuestro siguiente programa, APRENDE-SOLO, no necesita que se le vaya introduciendo la información a cucharaditas, como era el caso de GRAN ESFUERZO-X. Aquí lo tenemos en acción:

```
CUANTOS FACTORES? 3

INTRODUZCA FACTOR 1
? TIENE ALAS

INTRODUZCA FACTOR 2
? TIENE UN PAR DE OJOS

INTRODUZCA FACTOR 3
? COME GUSANOS

INTRODUZCA ELEMENTO 1
? GORRION

INTRODUZCA ELEMENTO 2
? HOMBRE
```

Una vez que tenemos esta información podemos empezar a ejecutar el programa; éste aprenderá por sí solo a diferenciar entre los varios objetos que le hemos programado:

```
AHORA LE DEMOSTRARE MI EXPERIENCIA...
PIENSE EN UNO DE LOS ELEMENTOS:
TIENE ALAS VERDAD? ('S' O 'N')
? N
                                       >  0
TIENE UN PAR DE OJOS VERDAD? ('S' O 'N')
? S
                                       >  1
COME GUSANOS VERDAD? ('S' O 'N')
? N
                                       >  0
                                  >  CEREBRO =   0
EL ELEMENTO ES: GORRION
ES ESTO CORRECTO? ('S' O 'N')
? N
```

```
AHORA LE DEMOSTRARE MI EXPERIENCIA...
PIENSE EN UNO DE LOS ELEMENTOS:
TIENE ALAS VERDAD? ('S' O 'N')
? S
                                       >  1
TIENE UN PAR DE OJOS VERDAD? ('S' O 'N')
? S
                                       >  1
COME GUSANOS VERDAD? ('S' O 'N')
? S
                                       >  1
                                  >  CEREBRO =    1
EL ELEMENTO ES: GORRION
ES ESTO CORRECTO? ('S' O 'N')
? S
```

Se observará que al principio se equivoca, pero después enseguida empieza a hacer hipótesis válidas:

```
AHORA LE DEMOSTRARE MI EXPERIENCIA...
PIENSE EN UNO DE LOS ELEMENTOS:
TIENE ALAS VERDAD? ('S' O 'N')
? S
                                       >  1
TIENE UN PAR DE OJOS VERDAD? ('S' O 'N')
? S
                                       >  1
COME GUSANOS VERDAD? ('S' O 'N')
? S
                                       >  1
                                  >  CEREBRO =   2
EL ELEMENTO ES: GORRION
ES ESTO CORRECTO? ('S' O 'N')
? S
```

```
AHORA LE DEMOSTRARE MI EXPERIENCIA...
PIENSE EN UNO DE LOS ELEMENTOS:
TIENE ALAS VERDAD? ('S' O 'N')
? N
                                 >  0
TIENE UN PAR DE OJOS VERDAD? ('S' O 'N')
? S
                                 >  1
COME GUSANOS VERDAD? ('S' O 'N')
? N
                                 >  0
                           >  CEREBRO =  -1
EL ELEMENTO ES: HOMBRE
ES ESTO CORRECTO? ('S' O 'N')
? S
```

En un determinado momento parece que se vuelve infalible:

```
AHORA LE DEMOSTRARE MI EXPERIENCIA...
PIENSE EN UNO DE LOS ELEMENTOS:
TIENE ALAS VERDAD? ('S' O 'N')
? S
                                 >  1
TIENE UN PAR DE OJOS VERDAD? ('S' O 'N')
? S
                                 >  1
COME GUSANOS VERDAD? ('S' O 'N')
? S
                                 >  1
                           >  CEREBRO =   1
EL ELEMENTO ES: GORRION
ES ESTO CORRECTO? ('S' O 'N')
? S
```

Cómo funciona

Lo más importante (y que supone además la mayor limitación) de este programa es que sólo puede distinguir entre dos elementos (como el HOMBRE y el GORRION de nuestro ejemplo). El programa se inicia suponiendo que su total (la variable CEREBRO) será mayor o igual a cero. El verdadero valor que CEREBRO llega a tener carece de importancia.

La primera vez que ejecutamos el programa, éste nos pide la información que va a necesitar:

```
390 REM INICIALIZACION
400 CLS
410 NELE = 2: REM NUMERO DE ELEMENTOS
420 PRINT: PRINT
430 INPUT "CUANTOS FACTORES"; FACTOR
440 DIM A$(NELE): REM NOMBRES DE LOS ELEMENTOS
450 DIM B$(FACTOR): REM NOMBRES DE LOS FACTORES
```

```
460 DIM C(FACTOR), D(FACTOR)
470 CLS
480 FOR J = 1 TO FACTOR
490 PRINT: PRINT
500 PRINT "INTRODUZCA FACTOR"; J
510 INPUT B$(J)
520 NEXT J
530 PRINT: PRINT
540 CLS
550 FOR J = 1 TO NELE
560 PRINT: PRINT
570 PRINT "INTRODUZCA ELEMENTO"; J
580 INPUT A$(J)
590 NEXT J
600 RETURN
```

Cada vez que pasa por el bucle, APRENDE-SOLO empieza rellenando con ceros cada elemento de la matriz C (hay un elemento para cada factor):

```
 30 REM *** BUCLE PRINCIPAL DE APRENDIZAJE ***
 40 CLS
 50 FOR J = 1 TO FACTOR
 60 C(J) = 0
 70 NEXT J
 80 PRINT
 90 GOSUB 130
100 GOTO 40
```

Seguidamente comienza a imprimir los factores, uno a uno, pidiéndonos que le digamos 'S' o 'N' si se refieren al elemento en el que hemos pensado:

```
120 REM TIEMPO DE DEMOSTRACION
130 PRINT "AHORA LE DEMOSTRARE MI EXPERIENCIA..."
140 PRINT "PIENSE EN UNO DE LOS ELEMENTOS:"
150 FOR J = 1 TO FACTOR
160 PRINT B$(J); " VERDAD? ('S' O 'N')"
170 INPUT Z$
180 IF Z$ <> "S" AND Z$ <> "N" THEN 170
190 IF Z$ = "S" THEN C(J) = 1
200 PRINT TAB(29); "> "; C(J)
210 NEXT J
```

Si decimos 'S' se fija a uno ese elemento de la matriz C. Una vez que hemos pasado este bucle, CEREBRO calcula un total para ese elemento, con su código comprendido entre las líneas 230 y 270:

```
220 CEREBRO = 0
230 FOR J = 1 TO FACTOR
240 CEREBRO = CEREBRO + C(J) * D(J)
250 NEXT J
260 PRINT TAB(25) ">  CEREBRO = "; CEREBRO
```

Si observamos atentamente este listado, vemos que la primera vez que se ejecuta este bucle, CEREBRO será igual a cero (porque todos los C(J) han sido multiplicados por los de D(J) y todos los D(J) empiezan siendo iguales a cero).

Esto significa que la primera vez que ejecutemos el programa, éste nos dará la opción uno (es decir, A$(1), el primer elemento que hemos introducido) si CEREBRO es igual a cero:

```
270 IF CEREBRO >= 0 THEN PRINT "EL ELEMENTO ES: "
    ; A$(1): EX = -1
280 IF CEREBRO < 0 THEN PRINT "EL ELEMENTO ES: "
    ; A$(2): EX = 1
```

APRENDE-SOLO nos pregunta entonces si tiene razón. Si le decimos que sí, no modifica esta información, porque —en su forma presente— dará la misma respuesta la próxima vez que se le presente la misma información. Sin embargo, si le decimos que está equivocado, atravesará el bucle siguiente, modificando los valores de D(J) y utilizando para ello los dos valores de C(J) que le dimos y la variable EX. Si retrocedemos a las líneas 270 y 280, veremos que EX se fija a —1 si el elemento en el que pensó era A$(1), y a 1 si pensó en el A$(2).

D(J) es el elemento vital de las líneas comprendidas entre la 230 y la 250 y ayuda a determinar el valor de CEREBRO; así pues, se ha de modificar si el programa produjo un resultado equivocado:

```
290 PRINT "ES ESTO CORRECTO? ('S' O 'N')"
300 INPUT Z$
310 IF Z$ <> "S" AND Z$ <> "N" THEN 300
320 PRINT
330 IF Z$ = "S" THEN 370
340 FOR J = 1 TO FACTOR
350 D(J) = D(J) + EX * C(J)
360 NEXT J
370 RETURN
```

Una vez que ha hecho los cambios oportunos a D(J), utilizando ambos valores de los elementos de la matriz C (que, como veremos en las líneas 60 y 200, sólo pueden tener

valores iguales a uno o cero), el programa vuelve para intentarlo de nuevo. Como veremos, enseguida resulta infalible.

Este es el listado completo (para usuarios del ZX Spectrum, véase Apéndice):

```
 10 REM APRENDE-SOLO
 20 GOSUB 390: REM INICIALIZACION
 30 REM *** BUCLE PRINCIPAL DE APRENDIZAJE ***
 40 CLS
 50 FOR J = 1 TO FACTOR
 60 C(J) = 0
 70 NEXT J
 80 PRINT
 90 GOSUB 130
100 GOTO 40
110 REM **********************
120 REM TIEMPO DE DEMOSTRACION
130 PRINT "AHORA LE DEMOSTRARE MI EXPERIENCIA..."
140 PRINT "PIENSE EN UNO DE LOS ELEMENTOS:"
150 FOR J = 1 TO FACTOR
160 PRINT B$(J); " VERDAD? ('S' O 'N')"
170 INPUT Z$
180 IF Z$ <> "S" AND Z$ <> "N" THEN 170
190 IF Z$ = "S" THEN C(J) = 1
200 PRINT TAB(29); "> "; C(J)
210 NEXT J
220 CEREBRO = 0
230 FOR J = 1 TO FACTOR
240 CEREBRO = CEREBRO + C(J) * D(J)
250 NEXT J
260 PRINT TAB(25) ">  CEREBRO = "; CEREBRO
270 IF CEREBRO >= 0 THEN PRINT "EL ELEMENTO ES: "
    ; A$(1): EX = -1
280 IF CEREBRO < 0 THEN PRINT "EL ELEMENTO ES: "
    ; A$(2): EX = 1
290 PRINT "ES ESTO CORRECTO? ('S' O 'N')"
300 INPUT Z$
310 IF Z$ <> "S" AND Z$ <> "N" THEN 300
320 PRINT
330 IF Z$ = "S" THEN 370
340 FOR J = 1 TO FACTOR
350 D(J) = D(J) + EX * C(J)
360 NEXT J
370 RETURN
380 REM **************
390 REM INICIALIZACION
400 CLS
410 NELE = 2: REM NUMERO DE ELEMENTOS
420 PRINT: PRINT
430 INPUT "CUANTOS FACTORES"; FACTOR
440 DIM A$(NELE): REM NOMBRES DE LOS ELEMENTOS
450 DIM B$(FACTOR): REM NOMBRES DE LOS FACTORES
460 DIM C(FACTOR), D(FACTOR)
470 CLS
480 FOR J = 1 TO FACTOR
```

```
490 PRINT: PRINT
500 PRINT "INTRODUZCA FACTOR"; J
510 INPUT B$(J)
520 NEXT J
530 PRINT: PRINT
540 CLS
550 FOR J = 1 TO NELE
560 PRINT: PRINT
570 PRINT "INTRODUZCA ELEMENTO"; J
580 INPUT A$(J)
590 NEXT J
600 RETURN
```

Más de dos alternativas

Aunque es fascinante el contar con un programa que aprende así, por sí solo, no deja de ser un gran inconveniente el hecho de que no es capaz de escoger entre más de dos alternativas. Nuestro siguiente programa, MULTI-APRENDE-SOLO, está pensado de modo que permite introducir más de dos elementos.

El programa empieza así (como era de esperar):

```
CUANTOS ELEMENTOS? 3

CUANTOS FACTORES? 3


POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 1
? HOMBRE
POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 2
? CABALLO
POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 3
? GORRION


POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 1
? SOLO TIENE UN PAR DE PATAS O PIERNAS
POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 2
? PUEDE VOLAR POR SI MISMO
POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 3
? RESPIRA OXIGENO


ESTOS SON LOS POSIBLES ELEMENTOS:

 HOMBRE
  CABALLO
   GORRION

POR FAVOR PIENSE EN UNO DE ELLOS

PULSE LA TECLA 'RETURN' CUANDO ESTE PREPARADO
?
```

No obstante, al ejecutarlo, veremos que el programa nos hace preguntas y luego intenta adivinar lo que estábamos pensando. Si se equivcca, nos pregunta cuál es la respuesta correcta:

```
POR FAVOR RESPONDA 'S' O 'N' A CADA UNA
DE LAS SIGUIENTES PREGUNTAS, RELACIONADAS
CON EL ELEMENTO EN QUE PENSO.

SOLO TIENE UN PAR DE PATAS O PIERNAS
? S
PUEDE VOLAR POR SI MISMO
? N
RESPIRA OXIGENO
? S
ESTABAS PENSANDO EN: CABALLO

INTRODUZCA 'S' SI TENGO RAZON, Y 'N' EN CASO CONTRARIO
? N
CUAL DEBERIA HABER SIDO LA RESPUESTA?
? HOMBRE
```

Cuando lo hayamos ejecutado un tiempo (que no será muy largo, con sólo tres elementos y tres factores), no fallará una:

```
POR FAVOR RESPONDA 'S' O 'N' A CADA UNA
DE LAS SIGUIENTES PREGUNTAS, RELACIONADAS
CON EL ELEMENTO EN QUE PENSO.

SOLO TIENE UN PAR DE PATAS O PIERNAS
? N
PUEDE VOLAR POR SI MISMO
? N
RESPIRA OXIGENO
? S
ESTABAS PENSANDO EN: CABALLO

INTRODUZCA 'S' SI TENGO RAZON, Y 'N' EN CASO CONTRARIO
? S


POR FAVOR RESPONDA 'S' O 'N' A CADA UNA
DE LAS SIGUIENTES PREGUNTAS, RELACIONADAS
CON EL ELEMENTO EN QUE PENSO.

SOLO TIENE UN PAR DE PATAS O PIERNAS
? S
PUEDE VOLAR POR SI MISMO
? S
RESPIRA OXIGENO
? S
ESTABAS PENSANDO EN: GORRION

INTRODUZCA 'S' SI TENGO RAZON, Y 'N' EN CASO CONTRARIO
? S
```

La parte más importante del programa se encuentra entre las líneas 150 y 520. La primera sección de éstas acepta las respuestas 'S' y aumenta una variable llamada CONTADOR dependiendo de la respuesta (añadiéndole 1 por la primera respuesta 'S', 2 por la segunda, 4 por la siguiente, luego 8, 16, 32 y así sucesivamente):

```
150 PRINT "POR FAVOR RESPONDA 'S' O 'N' A CADA
          UNA"
160 PRINT "DE LAS SIGUIENTES PREGUNTAS, RELACIO
          NADAS"
170 PRINT "CON EL ELEMENTO EN QUE PENSO."
180 PRINT
190 CCONTADOR = 0
200 X = .5
210 FOR J = 1 TO FACTOR
220 X = X + X
230 PRINT F$(J)
240 INPUT Z$
250 IF Z$ <> "S" AND Z$ <> "N" THEN 240
260 IF Z$ = "S" THEN CCONTADOR = CCONTADOR + X
270 NEXT J
```

Cuando el programa lleve un rato en ejecución, habrá asignado valores a muchos elementos de la matriz B. B(1) será el total cuando A$(1) es el elemento, B(2) será el total para un elemento de A$(6) y así sucesivamente. Después de haber dado las respuestas 'S' se atraviesa un pequeño bucle para ver si el total obtenido es igual a cualquiera de los valores B(J) previamente almacenados. De ser así, la variable X se fija igual a J:

```
280 X = 0
290 FOR J = 1 TO NELE
300 IF CCONTADOR = B(J) THEN X = J
310 NEXT J
```

Si se ha asignado este valor, X deja de ser igual a cero y el sistema toma una decisión:

```
320 IF X <> 0 THEN 410
```

Si no se obtiene aquí una respuesta definida, el ordenador genera un número aleatorio entre 1 y el número de elementos con el fin de lanzar una hipótesis. Pero esto no es suficiente para decir ESTABAS PENSANDO EN ";A$ (número aleatorio generado). Aunque puede que el programa todavía no haya asignado un A$(n) al total de n que hemos recibido, es posible que ya haya asignado algunos elementos de A$. Por tanto, puede y debe rechazar algunas de las hipótesis producidas por el generador de números aleatorios:

```
330 X = INT(RND(1) * NELE) + 1
340 REM ** RECHAZA TODAS LAS RESPUESTAS INCORREC
        TAS **
350 BANDERA = 0
360 FOR J = 1 TO NELE
370 IF B(J) = 0 THEN 390
380 IF X = J AND CCONTADOR <> B(J) THEN BANDERA =
    1
390 NEXT J
400 IF BANDERA = 1 THEN 330
```

En el caso de que la variable BANDERA sea igual a cualquier cosa excepto cero, no se puede usar la hipótesis (el elemento de A$ representado por el valor de X generado aleatoriamente), ya que el sistema ya sabe que la respuesta es errónea. Así pues, utilizando la línea 400, retrocede para encontrar un nuevo valor para X.

Una vez hecho esto, el programa comprueba su hipótesis:

```
410 PRINT "ESTABAS PENSANDO EN: "; A$(X)
420 PRINT
430 PRINT "INTRODUZCA 'S' SI TENGO RAZON, Y 'N'
           EN CASO CONTRARIO"
440 INPUT Z$
450 IF Z$ <> "S" AND Z$ <> "N" THEN 440
460 IF Z$ = "S" THEN B(X) = CCONTADOR: GOTO 30
470 PRINT "CUAL DEBERIA HABER SIDO LA RESPUESTA?"
480 INPUT Z$
490 FOR J = 1 TO NELE
500 IF A$(J) = Z$ THEN B(J) = CCONTADOR
510 NEXT J
520 GOTO 30
```

Si ésta era correcta, el sistema usa el bucle comprendido entre las líneas 490 y 510 para descubrir qué elemento de A$ corresponde al total generado en esa ejecución. Esto garantiza que la próxima vez que se encuentre con ese total será capaz de identificar el elemento relevante de A$.

Este es el listado completo que nos permite construir un sistema experto que aprenda por sí solo (para usuarios del ZX Spectrum, véase Apéndice):

```
10 REM MULTI-APRENDE-SOLO
20 GOSUB 550: REM INICIALIZACION
30 CLS
40 PRINT "ESTOS SON LOS POSIBLES ELEMENTOS:"
50 PRINT
60 FOR J = 1 TO NELE
```

```
70 PRINT TAB(J); A$(J)
80 NEXT J
90 PRINT
100 PRINT "POR FAVOR PIENSE EN UNO DE ELLOS"
110 PRINT
120 PRINT "PULSE LA TECLA 'RETURN' CUANDO ESTE
          PREPARADO"
130 INPUT Z$
140 CLS
150 PRINT "POR FAVOR RESPONDA 'S' O 'N' A CADA
          UNA"
160 PRINT "DE LAS SIGUIENTES PREGUNTAS, RELACIO
          NADAS"
170 PRINT "CON EL ELEMENTO EN QUE PENSO."
180 PRINT
190 CCONTADOR = 0
200 X = .5
210 FOR J = 1 TO FACTOR
220 X = X + X
230 PRINT F$(J)
240 INPUT Z$
250 IF Z$ <> "S" AND Z$ <> "N" THEN 240
260 IF Z$ = "S" THEN CCONTADOR = CCONTADOR + X
270 NEXT J
280 X = 0
290 FOR J = 1 TO NELE
300 IF CCONTADOR = B(J) THEN X = J
310 NEXT J
320 IF X <> 0 THEN 410
330 X = INT(RND(1) * NELE) + 1
340 REM ** RECHAZA TODAS LAS RESPUESTAS INCORREC
       TAS **
350 BANDERA = 0
360 FOR J = 1 TO NELE
370 IF B(J) = 0 THEN 390
380 IF X = J AND CCONTADOR <> B(J) THEN BANDERA =
    1
390 NEXT J
400 IF BANDERA = 1 THEN 330
410 PRINT "ESTABAS PENSANDO EN: "; A$(X)
420 PRINT
430 PRINT "INTRODUZCA 'S' SI TENGO RAZON, Y 'N'
          EN CASO CONTRARIO"
440 INPUT Z$
450 IF Z$ <> "S" AND Z$ <> "N" THEN 440
460 IF Z$ = "S" THEN B(X) = CCONTADOR: GOTO 30
470 PRINT "CUAL DEBERIA HABER SIDO LA RESPUESTA?"
480 INPUT Z$
490 FOR J = 1 TO NELE
500 IF A$(J) = Z$ THEN B(J) = CCONTADOR
510 NEXT J
520 GOTO 30
530 END
540 REM **************
550 REM INICIALIZACION
560 CLS
570 INPUT "CUANTOS ELEMENTOS"; NELE
```

```
580 PRINT
590 INPUT "CUANTOS FACTORES"; FACTOR
600 CLS
610 X = NELE + FACTOR
620 DIM A$(NELE)
630 DIM F$(FACTOR), B(X)
640 FOR J = 1 TO NELE
650 PRINT "POR FAVOR INTRODUZCA EL NOMBRE DEL
          ELEMENTO"; J
660 INPUT A$(J)
670 NEXT J
680 CLS
690 FOR J = 1 TO FACTOR
700 PRINT "POR FAVOR INTRODUZCA EL NOMBRE DEL
          FACTOR"; J
710 INPUT F$(J)
720 NEXT J
730 RETURN
```

Sin corrección

El último programa de esta sección es una variación del que acabamos de ver. La única diferencia entre aquél y el que vamos a ver ahora es que este último no necesita que se le diga cuál hubiera sido la respuesta correcta en el caso de que se equivoque. El programa descubre solo y bastante rápidamente cuál es el modo de distinguir entre los diferentes elementos.

No he vuelto a numerar este programa, de modo que es muy fácil modificar el anterior convirtiéndolo en MULTI-APRENDE-SOLO 2. Esta es una ejecución del programa:

```
             CUANTOS ELEMENTOS? 5

             CUANTOS FACTORES? 5

POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 1
? PERRO
POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 2
? CABALLO
POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 3
? OVEJA
POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 4
? CUERVO
POR FAVOR INTRODUZCA EL NOMBRE DEL ELEMENTO 5
? HOMBRE


POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 1
? PUEDE HABLAR
POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 2
? LADRA
POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 3
? PUEDE VOLAR POR SI MISMO
POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 4
? TIENE CUATRO PATAS
POR FAVOR INTRODUZCA EL NOMBRE DEL FACTOR 5
? PRODUCE LANA
```

Una vez que tiene situada la base de datos, el programa procede del modo siguiente:

```
ESTOS SON LOS POSIBLES ELEMENTOS:

PERRO
 CABALLO
  OVEJA
   CUERVO
    HOMBRE

POR FAVOR PIENSE EN UNO DE ELLOS

PULSE LA TECLA 'RETURN' CUANDO ESTE PREPARADO
? •
```

```
POR FAVOR RESPONDA 'S' O 'N' A CADA UNA
DE LAS SIGUIENTES PREGUNTAS, RELACIONADAS
CON EL ELEMENTO EN QUE PENSO.

PUEDE HABLAR
? N
LADRA
? S
PUEDE VOLAR POR SI MISMO
? N
TIENE CUATRO PATAS
? S
PRODUCE LANA
? N
ESTABAS PENSANDO EN: CUERVO

INTRODUZCA 'S' SI TENGO RAZON, Y 'N' EN CASO CONTRARIO
? N
```

Aunque al principio de la ejecución la mayoría de las hipótesis serán erróneas, a medida que ésta avance, el programa irá acertando cada vez con mayor frecuencia:

```
POR FAVOR RESPONDA 'S' O 'N' A CADA UNA
DE LAS SIGUIENTES PREGUNTAS, RELACIONADAS
CON EL ELEMENTO EN QUE PENSO.

PUEDE HABLAR
? N
LADRA
? N
PUEDE VOLAR POR SI MISMO
? S
TIENE CUATRO PATAS
? N
PRODUCE LANA
? N
ESTABAS PENSANDO EN: CUERVO

INTRODUZCA 'S' SI TENGO RAZON, Y 'N' EN CASO CONTRARIO
? S
```

Al final no fallará una. Funciona de un modo muy parecido al programa anterior, asignando valores a los elementos de la matriz B. No obstante, en esta ocasión el programa cuenta con una matriz, C, que garantiza que nunca hará la misma hipótesis errónea partiendo del mismo total. La función de esta matriz es ir limitando cada vez más el número de elementos posibles, de modo que cada vez serán menos entre los que tenga que adivinar. No le lleva mucho tiempo al programa el construir una "visión del mundo" que le garantice que siempre estará en lo cierto. El número de veces que tenga que probar para llegar a no equivocarse nunca depende, como era de esperar, del número de elementos.

Una vez que el programa se las ha arreglado para aprender a distinguir por sí solo y sin cometer errores entre los diferentes elementos, contará con una base de datos a la que referirse en las pruebas siguientes. Esta es la base que MULTI-APRENDE-SOLO construyó durante esta ejecución:

J = 1	B(J) = 10	A$(J) = PERRO
J = 2	B(J) = 8	A$(J) = CABALLO
J = 3	B(J) = 24	A$(J) = OVEJA
J = 4	B(J) = 4	A$(J) = CUERVO
J = 5	B(J) = 1	A$(J) = HUMANO

Y éste es el listado (para usuarios del ZX Spectrum, véase Apéndice):

```
 10 REM MULTI-APRENDE-SOLO-2
 20 GOSUB 550: REM INICIALIZACION
 30 CLS
 40 PRINT "ESTOS SON LOS POSIBLES ELEMENTOS:"
 50 PRINT
 60 FOR J = 1 TO NELE
 70 PRINT TAB(J); A$(J)
 80 NEXT J
 90 PRINT
100 PRINT "POR FAVOR PIENSE EN UNO DE ELLOS"
110 PRINT
120 PRINT "PULSE LA TECLA 'RETURN' CUANDO ESTE
           PREPARADO"
130 INPUT Z$
140 CLS
150 PRINT "POR FAVOR RESPONDA 'S' O 'N' A CADA
           UNA"
160 PRINT "DE LAS SIGUIENTES PREGUNTAS, RELACIO
           NADAS"
170 PRINT "CON EL ELEMENTO EN QUE PENSO."
180 PRINT
190 CCONTADOR = 0
200 X = .5
210 FOR J = 1 TO FACTOR
220 X = X + X
230 PRINT F$(J)
240 INPUT Z$
250 IF Z$ <> "S" AND Z$ <> "N" THEN 240
```

```
260 IF Z$ = "S" THEN CCONTADOR = CCONTADOR + X
270 NEXT J
280 X = 0
290 FOR J = 1 TO NELE
300 IF CCONTADOR = B(J) THEN X = J
310 NEXT J
320 IF X <> 0 THEN 410
330 X = INT(RND(1) * NELE) + 1
340 REM ** RECHAZA TODAS LAS RESPUESTAS INCORREC
        TAS **
350 BANDERA = 0
360 FOR J = 1 TO NELE
370 IF B(J) = 0 THEN 390
380 IF X = J AND CCONTADOR <> B(J) THEN BANDERA =
    1
385 IF C(X) = CCONTADOR OR D(X) = CCONTADOR THEN
    BANDERA = 1
390 NEXT J
400 IF BANDERA = 1 THEN 330
410 PRINT "ESTABAS PENSANDO EN: "; A$(X)
420 PRINT
430 PRINT "INTRODUZCA 'S' SI TENGO RAZON, Y 'N'
           EN CASO CONTRARIO"
440 INPUT Z$
450 IF Z$ <> "S" AND Z$ <> "N" THEN 440
460 IF Z$ = "S" THEN B(X) = CCONTADOR: GOTO 30
470 IF C(X) = 0 THEN C(X) = CCONTADOR: GOTO 30
520 D(X) = CCONTADOR: GOTO 30
530 END
540 REM **************
550 REM INICIALIZACION
560 CLS
570 INPUT "CUANTOS ELEMENTOS"; NELE
580 PRINT
590 INPUT "CUANTOS FACTORES"; FACTOR
600 CLS
610 X = NELE + FACTOR
620 DIM A$(NELE)
630 DIM F$(FACTOR), B(X), C(X), D(X)
640 FOR J = 1 TO NELE
650 PRINT "POR FAVOR INTRODUZCA EL NOMBRE DEL
           ELEMENTO"; J
660 INPUT A$(J)
670 NEXT J
680 CLS
690 FOR J = 1 TO FACTOR
700 PRINT "POR FAVOR INTRODUZCA EL NOMBRE DEL
           FACTOR"; J
710 INPUT F$(J)
720 NEXT J
730 RETURN
```

Apéndices

1

Algunos consejos para mejorar las técnicas de programación

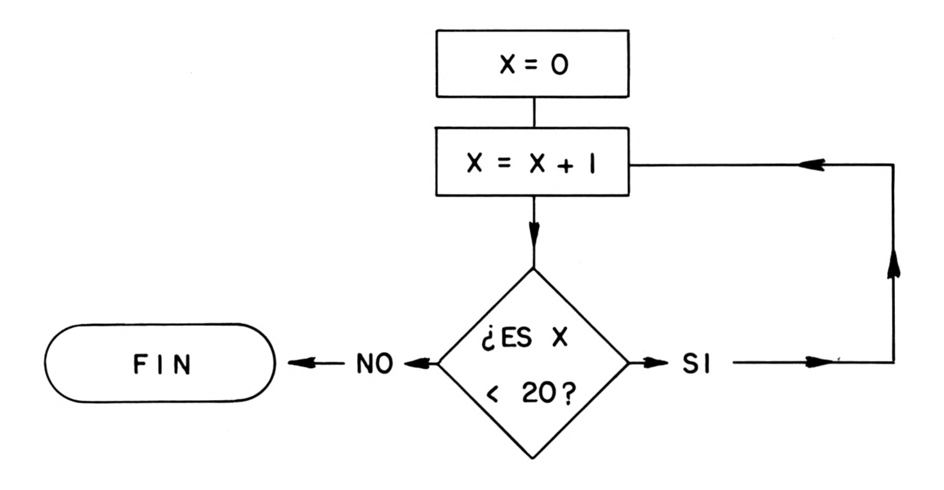
En nuestro desarrollo como programadores siempre acabamos llegando a ese punto en que, dominando ya una gran parte del lenguaje que estamos utilizando, podemos concentrarnos en escribir mejores programas: unos programas que funcionen con relativamente poca corrección *(debuggins),* que sean fáciles de entender y operar por otras personas y que estén escritos con lógica y elegancia. Los programas de inteligencia artificial suelen resultar especialmente diabólicos por su complejidad; así pues, todo lo que podamos hacer para facilitar su comprensión, será siempre muy bien recibido.

Hay más posibilidades de que un programa pueda ejecutarse la primera vez, cuando ha sido cuidadosamente planificado antes de empezar a introducir código (es decir, programa) en el ordenador.

Una buena manera de empezar es utilizar un esquema que se suele llamar ''diagrama de flujo''. Un diagrama de flujo es una serie de cuadrados y otras formas unidas por líneas que muestra el curso de la acción y la toma de decisiones en el ordenador cuando se ejecute el programa.

Las formas que se usen no son demasiado importantes; yo sugiero que se utilicen dos: un rectángulo para la mayoría de las acciones que ha de llevar a cabo el ordenador y una forma romboidal para las decisiones que tome el ordenador. Las esquinas de esta forma romboidal se pueden utilizar, como se ve en el siguiente diagrama, para atender a las alternativas con la que ha de enfrentarse el ordenador.

Este diagrama muestra el diagrama de flujo de un programa que fija la variable X igual a cero. Se comprueba el valor de X. Si se encuentra que es menor de 20, el programa retrocede para volver a añadirle uno a X. El proceso continúa hasta que el valor de X es igual a 20.



Una de las ventajas de utilizar un diagrama de flujo es que no te ves atado, nada más empezar, por las peculiaridades del lenguaje que estás utilizando, sus puntos débiles y fuertes. En su lugar, uno puede concentrarse en lo que quiere hacer. Por supuesto, lo más seguro es que tenga que modificar ligeramente sus planes para aceptar las limitaciones impuestas por el lenguaje de la programación, sea el que fuere, pero no se tendrá que doblegar a estas limitaciones (que, en parte, pueden ser imaginarias) al principio del proceso.

El diagrama de flujo es universal. Puede usarse el mismo diagrama como base de un programa escrito en un ordenador equipado con un lenguaje totalmente diferente de aquel con el que, en principio, se pensaba escribir el programa.

Un diagrama de flujo modela el curso (o flujo) de la acción y la lógica del programa siendo, por tanto, muy útil para descubrir errores nada más empezar a escribirlo. Por ejemplo, se puede encontrar que una determinada condición no se va a cumplir nunca, lo que puede llevar a que el programa termine atrapado en un bucle infinito. Otras partes del código pueden desviarse completamente porque la condición que dispara la entrada en esa parte del código tampoco llega a satisfacerse nunca.

El siguiente paso, tras haber diseñado un diagrama de flujo para el programa y tras haberlo ejecutado *in mente* unas cuantas veces para corregir los errores más obvios, es reducir el diagrama de flujo a una serie de llamadas de subrutinas. Aunque parece un poco tonto hacerlo en el caso de un programa como éste: "FIJA X IGUAL A CERO, AÑADE UNO, COMPRUEBA SI ES MENOR DE VEINTE", este método se muestra particularmente valioso con programas más complejos, como los que intentan demostrar la inteligencia artificial.

Programación en módulos

Como se habrá observado en casi todos los programas que componen este libro, yo soy partidario de iniciar cada programa con una serie de llamadas a la subrutina, ocupándose cada una de ellas por separado de realizar las diferentes acciones. Con esto, si

en un programa determinado hubiera que realizar varias veces una secuencia determinada, se puede llamar reiteradamente a la subrutina hasta que se satisfaga una determinada condición que señale el final de la ejecución.

El lector se dará cuenta de lo útil que resulta este modo de programación cuando llegue a la etapa en la que se corrigen los errores que pueda tener el programa. De haber un error, lo más probable es que éste se dé en una sola subrutina, de modo que no le resultará demasiado difícil localizar cuál es la que lo contiene, en vez de tener que abrirse paso por todo el programa intentando seguir las huellas de dicho error.

Cuando se trabaja con módulos de subrutinas, uno puede ir comprobando por separado las diferentes secciones del programa, incluso antes de que éste esté totalmente finalizado. Intentaré dejar esto bien claro mostrando la primera parte de un programa típico para jugar a las DAMAS (de hecho, se comprobará que el programa SNICKERS, incluido en este libro, empieza también de un modo muy parecido a éste). Nuestro programa de DAMAS ideal empezaría así:

```
 10 REM COMPROBACION
 20 GOSUB 9000: REM INICIALIZA LAS VARIABLES
 30 GOSUB 8000: REM IMPRESION DEL TABLERO
 40 GOSUB 7000: REM ACEPTA EL MOVIMIENTO DEL
    JUGADOR
 50 GOSUB 8000: REM IMPRESION DEL TABLERO
 60 GOSUB 5000: REM EL ORDENADOR REALIZA UN
    MOVIMIENTO
 70 IF (la persona no ha ganado) AND (el ORde
    nador no ha ganado) THEN 30
 80 IF (el ORdenador ha ganado) THEN PRINT "HE
    GANADO"
 90 IF (la persona ha ganado) THEN PRINT "HAS
    GANADO"
100 END
```

Se podría empezar a ejecutar y comprobar una parte bastante apreciable del programa (como la rutina de iniciación, la de impresión del tablero y la que acepta la jugada del jugador humano) antes incluso de ponernos a pensar sobre cómo demonios vamos a conseguir que el ordenador haga su jugada.

Sabríamos, por ejemplo, que no tendríamos que perder más tiempo pensando si se debe o no a un error en la rutina de impresión del tablero el que la salida sea un poco rara. Habiendo comprobado la subrutina del tablero y la de la jugada del jugador humano, sabemos que el error debe estar en la subrutina comprendida entre las líneas 5000 y 6999, la rutina en la que el ordenador lleva a cabo sus jugadas.

Todo lo que hay que hacer es poner una sola sentencia PRINT, como "JUGADA DEL ORDENADOR", seguida de un RETURN para subrutinas incompletas, sabiendo que el programa va a aceptarlo, y demostrar la dirección que sigue el curso del programa, aun cuando todavía no hayamos escrito todas las secciones de código.

Como norma general, yo aconsejo a mis lectores que utilicen el sistema que implica el

uso de un ''bucle principal'' de llamadas a la subrutina dentro del cual podrán ''contener'' el programa entero.

También sugiero que se tenga escrito el programa, o una gran parte de éste, antes de encender el ordenador; ya sé que para ello es preciso vencer la gran tentación de sumergirse directamente en el mundo de la máquina y empezar a perforar secciones de código a medida que se nos van ocurriendo, pero merece la pena imponerse cierta disciplina en este sentido. Quien se decida a hacerlo descubrirá enseguida lo útil que resulta el escribir primero el programa a mano y, además, a la larga, los resultados que obtenga serán mucho mejores que en el caso de no haberlo escrito primero.

Y, sobre todo, siguiendo este procedimiento, acabará tardando menos tiempo que si hubiera empezado el proceso sentado ante el teclado del ordenador.

A mí me llevó un tiempo aprender esta lección. Aunque había leído muchas veces consejos del tipo de ''calcule exactamente lo que va a hacer antes de empezar a introducir información en el ordenador'', solía lanzarme directamente a la máquina sin muchas ideas previas.

Aunque hacía un diagrama de flujo y tenía una idea del tipo de organización que quería conseguir en la pantalla, es cierto que no solía escribir el programa en papel antes de empezar a teclearlo en el ordenador. Pero entonces, en una ocasión me sucedió que, bulléndome mil ideas en la cabeza no para uno, sino para varios programas, no tenía, sin embargo, la posibilidad de acceder a ordenador alguno y me vi obligado a escribirlos en un cuaderno.

La relativa facilidad con la que pude corregir los programas cuando finalmente los introduje en un ordenador y la complejidad de los mismos (entre los que se incluye mi primer programa de AJEDREZ), me convencieron de que ésta era la manera de trabajar y así lo hice en lo sucesivo.

Cuando trabajo en programas de inteligencia artificial, suelo escribir en primer lugar el ''bucle de llamada de subrutinas'', pero sin numerar las líneas, de modo que el programa contiene líneas como GOSUB PRINT BOARD y GOSUB INITIALISE. Seguidamente, escribo cada módulo (o subrutina) en una hoja de papel diferente y luego, cuando ya he escrito todas las subrutinas más importantes, las ordeno como me parece más lógico.

Todo esto ocurre, por supuesto, antes de escribir los números de las líneas. Los módulos de las subrutinas se ordenan de un modo que garantice que la estructura del programa sea lo más lógica posible. Yo utilizo flechas para indicar el destino de los GOTO dentro de un módulo y nombres para la subrutina, como una sugerencia en el bucle principal. Posteriormente, cuando el programa va tomando forma, numero las líneas (siempre lo hago de diez en diez, empezando en la línea 10) y añado los destinos relevantes de los GOTO y GOSUB.

Todos los programas cuentan con una ''condición final'', punto en el cual finaliza el proceso. Merece la pena poner una comprobación para esta condición; dicha comprobación puede formar parte del GOTO que hace retroceder el programa de modo que vuelve a ejecutarse. Esto garantiza que el ciclo continuará hasta que se cumpla una determinada condición, momento en el que el programa se desvía de ese GOTO y continúa con las líneas que indican el fin del programa.

Signos de entrada explícitos y sentencias PRINT

Cuando escribimos un programa debemos tener en mente cómo lo va a ver un extraño cuando lo ejecute por primera vez. Si es necesario introducir un signo de entrada, es mucho más útil que el programa imprima algo como "¿CUANTAS HORAS HA TRABAJADO EL EMPLEADO ESTA SEMANA?", en vez de imprimir simplemente "INTRODUZCA HORAS?" o, lo que es peor, porque no serviría de nada, un signo de interrogación.

La misma sugerencia puede aplicarse a la impresión de la salida. Es mucho mejor que el programa imprima "EL NUMERO DE HORAS TRABAJADAS ESTA SEMANA ES 27", en vez de "HORAS: 27" o simplemente "27". Es cierto que el proporcionar signos de entrada y sentencias PRINT explícitas consume memoria, al tiempo que se ha de pensar también en el tiempo que lleva el teclearlas al introducir el programa, pero la contribución que éstas aportan al programa final significa que merecen la pena, pese a los problemas que al principio puedan plantear.

Sentencias REM

Mientras que las sentencias PRINT y los signos de entrada explícitos ayudarán a comprender el programa a la persona que vaya a ejecutarlo, las sentencias REM pueden ayudar a entender el listado a aquellos que lo lean por primera vez. Las sentencias REM (que, como todos sabemos, serán totalmente ignoradas por el ordenador a la hora de ejecutar el programa) deben usarse para iluminar el curso lógico del programa y aclarar lo que sucede en ciertas secciones de éste. Son esencialmente importantes en aquellas partes del programa en las que se toman decisiones o se realizan cálculos.

Las sentencias REM no sólo pueden utilizarse para explicar las diferentes partes del código, sino también para proporcionar "cortes" visuales, de modo que los diferentes bloques de código que llevan a cabo ciertas tareas están visualmente separados del resto del programa. Para ello se puede utilizar una sentencia REM en blanco (es decir, la palabra REM sola). También se puede utilizar una línea de asteriscos.

Variables

Merece la pena que comentemos un poco el uso de los nombres explícitos para las variables, ya sea con la palabra entera (tal que HORAS, como nombre variable, en un programa que se encargue de las nóminas de una empresa, para indicar horas trabajadas) o abreviada (como HR), siempre que su significado sea obvio. Esto facilita la tarea de seguir el rastro de las variables cuando se trabaja en un programa determinado (como el que acabamos de mencionar). Es muy útil cuando se realiza la primera corrección *(debugging)* y después también, cuando se haga necesario mejorar o ampliar el programa.

Asimismo, los nombres variables explícitos ayudan a clarificar el código; de este modo, cuando otros programadores quieran calcular cuál es la misión de las diferentes partes del programa, no encontrarán grandes dificultades. Por otro lado, suponen una gran ayuda para el propio programador del programa cuando éste vuelva a retomarlo

después de pasado un tiempo. Es sorprendente cómo un código que parece sorprendentemente claro, en términos de su objetivo, cuando uno lo introduce al hacer el programa, puede resultar denso y difícil de seguir cuando uno vuelve al programa después de mucho tiempo.

Comprobación de la entrada

El programa debe comprobar toda la entrada que introduzca el usuario antes de aceptarla, lo que garantiza que éste no va a venirse abajo en un momento determinado debido a datos incorrectos. Ya se quiera utilizar una entrada en cadena alfanumérica o una entrada numérica, suele ser más acertado utilizar la entrada en cadena alfanumérica, que se comprueba primero para ver si el material introducido es aceptable y luego, de ser necesario, la cadena puede sustituirse por un número.

Por ejemplo, si el usuario necesita introducir un número entre el uno y el nueve, puede aceptarse una cadena y después comprobar para asegurarnos de que no es menor a "1" ni mayor a "9" antes de sustituirlo por un número con una orden del tipo VAL.

Al tiempo que hemos de asegurarnos de que el programa rechazará toda entrada que no sea válida, tenemos que comprobarlo también verificando que todas las entradas que éste acepta producirán, cuando sean procesadas posteriormente, las respuestas adecuadas. Por ejemplo, debemos asegurarnos de que el programa no acepta el cero como un número posible, si el ordenador ha de dividir después por ese número.

Del mismo modo, si se han de procesar números mediante una función y luego usar para una división el resultado de ese procesado, se ha de verificar que una entrada aparentemente válida no se convierte en cero como resultado de la evaluación realizada por la función.

Si el ordenador rechaza la entrada del usuario y le pide una nueva, el programa, idealmente hablando, debería señalar por qué no era válida la entrada anterior o volver a imprimir exactamente qué es lo que se requiere (como "INTRODUZCA UN NUMERO ENTRE 1 y 4"). Nos arriesgamos a que todos los que utilicen nuestro programa acaben echando chispas contra nosotros si cada vez que introducen una entrada aparentemente válida, el programa se la rechaza sin darle una razón.

Documentación

Se suele llamar documentación a todo el material escrito que acompaña a un programa. Por muy reducida que sea, esta documentación suele resultar muy útil como apoyo para el programa.

La información escrita debe explicar, por supuesto, lo que hace el programa y después bosquejar el curso de la acción de éste. La documentación debe avisar al usuario sobre lo que debe hacer cuando esté ejecutando el programa, además de indicar los tipos de entrada que serán aceptados por el programa.

También se debe comentar en esta documentación el formato de la salida final. Se puede incluir una lista de nombres variables.

Asimismo, se pueden añadir los medios, de haber alguno, de desarrollar, ampliar o

mejorar el programa. Al igual que referencias escritas relativas a cualquier material que pueda ayudar o comprender los algoritmos utilizados en el programa o que sugiera áreas para el desarrollo del mismo.

Es razonable considerar, desde diferentes puntos de vista, que la tarea de programar no termina cuando finalizamos el programa. Sin una documentación adecuada, la tarea no puede darse por acabada sino en sus tres cuartas partes. La documentación añade un sello de profesionalidad al trabajo, lo que permite, a su vez, que el programa pueda usarse de un modo más efectivo.

Posiblemente, el único caso en que no es necesaria una documentación muy extensiva es cuando el programa está totalmente especificado mediante las sentencias REM. Un programa que utilice gran cantidad de sentencias de este tipo puede no necesitar documentación alguna, especialmente si se incluye una lista de variables, a modo de una serie de sentencias REM, al final del programa.

No obstante, por lo general, suele ser preferible documentar externamente el programa, en vez de confiar en que las sentencias REM, o las diferentes posibilidades ofrecidas, lo hagan por nosotros. Cuando se escribe la documentación de un programa se debe tener en cuenta que va dirigido a alguien que quizá nunca haya visto una ejecución de dicho programa.

La persona que la lea ha de hacerse una idea lo más exacta posible de qué es lo que hace el programa y cómo lo hace; cómo actúa recíprocamente con el usuario, tanto a la hora de aceptar la información que éste le introduce, como a la de presentarle los resultados de sus cómputos; y cómo está organizado en su conjunto.

Cuando escribimos un programa de inteligencia artificial, suele ser muy útil incluir en la documentación alguna información relativa a la naturaleza de la inteligencia artificial que va a intentar demostrar el programa. Se pueden dar referencias de artículos o libros que traten en particular sobre ese campo. De este modo, el usuario podrá apreciar, además, el buen trabajo que hemos hecho.

La documentación de un programa de envergadura ha de empezar con una introducción que explique rápidamente lo que sucede en el programa y cómo se usa éste, para posteriormente pasar a comentarlo en más detalle. No es un buen camino el forzar al usuario a abrirse paso a través de una maraña de información con el fin de sacar los hechos vitales, necesarios para empezar a ejecutar el programa.

2
Sugerencias bibliográficas

La bibliografía relativa a la Inteligencia Artificial es ya muy abundante y además no deja de crecer de día en día. La mayoría de las obras sobre el tema por mí consultadas merecen la pena, por el momento hay poca morralla.

No pretendo ofrecer en este apéndice una relación exhaustiva de toda la bibliografía existente. Más bien comentaré los libros que he encontrado particularmente interesantes y valiosos. Reseño cada uno de ellos con cierto detalle, de modo que el lector pueda hacerse una idea de lo que contienen.

He incluido algunos libros que sólo tocan el tema de la inteligencia artificial de un modo marginal (como los libros de Levy sobre el AJEDREZ y las DAMAS o la Antología de Asimov) porque los temas que tratan son interesantes para cualquiera que trabaje en este campo. Asimismo, sugieren áreas de investigación interesantes.

ARE COMPUTERS ALIVE?, Geoff Simons (Brighton Sussex, The Harvester Press Ltd., 1983).

Se trata de una visión de conjunto sobre la situación actual en la investigación de la inteligencia artificial; es muy fácil de leer y pone un énfasis especial en los robots. Cubre una gran parte del campo, pero sin que por ello quede sacrificada la profundidad necesaria para situar las cosas en su contexto histórico.

ARTIFICIAL INTELLIGENCE, Patrick Henry Winston (Reading, Massachusetts, Addison-Wesley Publishing Co., 1984).

Winston es profesor de Informática y también director del Laboratorio de Inteligencia Artificial en MIT. Esta es la segunda edición de un libro que, tras su publicación en 1977, no tardó en convertirse en una obra de consulta tradicional en este campo.

Winston señala en esta edición que en los siete años transcurridos desde que lo escribió, han cambiado muchas cosas en el campo de la inteligencia artificial. El cambio fundamental, dice, radica en el paso dado de los "ejemplos" de inteligencia artifical (programas que muestran lo que se podría hacer) a las "demostraciones" que expresan de modo real algunas de las promesas del pasado (sistemas de inteligencia artificial prácticos en funcionamiento).

Entre el mucho material que incluye, este libro ofrece un gran número de ejemplos sobre las limitaciones prácticas que han de reconocerse cuando se trabaja en el campo de la inteligencia artificial. Winston señala que, para comprender la inteligencia artificial y trabajar en este dominio, se han de comprender primero algunas ideas básicas. Estas son: equivalencias, reducción del objetivo, explotación de la restricción, búsqueda, control, solución de problemas y lógica.

ARTIFICIAL INTELLIGENCE, An Introductry Course, A Bundy (Edinburgo, Edinburgh University Press, 1980).

Se trata de los apuntes recogidos en un curso sobre inteligencia artificial. El curso centra su atención en la construcción de programas, más que en dar a los alumnos una visión general del campo. El material ofrecido sobre la manipulación de bloques y la investigación mediante árboles es particularmente interesante.

El libro está dividido en seis partes: Representación del Conocimiento, Lenguaje natural, Respuesta de preguntas y deducción, Percepcción visual, Aprendizaje y Programación. Se ofrecen también algunos programas escritos en LOGO. El estilo del libro es bastante seco, dados los fines para los que fue escrito, pero no por ello deja de ser menos valioso. También se incluyen traducciones a LISP de la mayoría de los programas ofrecidos en el libro.

ARTIFICIAL INTELLIGENCE, An MIT Perspective, P. H. Winston y R. H. Brown (Cambridge, Massachusetts, The MIT Press, 1980).

Este libro, que consta de dos volúmenes, es un densísimo tratado teórico, lleno de fórmulas y expresiones lógicas. Es una obra de un gran valor siempre que uno pueda desenvolverse a través de todos estos símbolos arcanos.

ARTIFICIAL VISION FOR ROBOTS, I. Aleksander (Londres, Kogan Page Ltd., 1985).

Es una visión de conjunto fascinante de todo el campo de la inteligencia artificial. Aleksander es profesor de Electrónica en la Brunnel University, pero su introducción al libro es clara y fácil de leer, así como sus otras dos contribuciones a la obra.

El último capítulo es una de ellas y en éste Aleksander empieza haciendo un rápido repaso a la situación actual de la investigación en el campo de la inteligencia artificial y luego pasa a comentar las "propiedades emergentes". Esta frase proviene de la descripción del comportamiento inteligente de un animal como la propiedad emergente del tejido nervioso que compone su cerebro. Dice que se podría argumentar que el comportamiento inteligente de un ordenador no es una propiedad emergente de la máquina (no más de lo que la pintura lo es del lienzo), pero, sin embargo, se podría considerar que la inteligencia artificial es una propiedad emergente de las reglas sobre las que está escrito cada programa en particular y que merece la pena seguir esa línea de investigación. Por

todo, es un libro muy interesante que muestra claramente cómo se sobreponen los diferentes campos de la inteligencia artificial.

AUTOMATIC NATURAL LANGUAGE PARSING, K. Sparck Jones y Y. Wilks (Chichester, West Sussex, Ellis Horwood Ltd., y Nueva York, Halsted Press, 1983).

Este libro estudia las tendencias actuales en el campo de los problemas y técnicas del análisis automático de una lengua natural. Este libro señala que el análisis gramatical es el elemento clave del procesado del lenguaje natural en su conjunto y es vital en áreas tales como las investigaciones de orientación lingüística y con los subsistemas que forman parte de los sistemas expertos.

BUILD YOUR OWN EXPERT SYSTEM, Chris Naylor (Cheshiere, Sigma Technical Press, 1983).

Este libro se ganó rápidamente la fama de ser la mejor guía existente en el mercado para la construcción de sistemas expertos. Cuando empecé a leerlo me quedé perplejo al descubrir que Chris intentaba ser gracioso. Después descubrí que el material que ofrecía era bueno y el humor ciertamente ayuda a pasar un texto bastante denso.

La primera parte del libro es razonablemente fácil de entender y los listados iluminan el texto o, para ser más exacto, proporcionan de hecho algunos programas expertos de gran mérito. Después se va haciendo más denso y complicado y requiere una lectura detenida, sobre todo cuando se llega al Teorema de Bayes u otros temas similares, pero todo ello está bien explicado, de modo que una lectura tenaz puede iluminar incluso el texto más complicado.

Para cerrar el libro, Naylor proporciona una gran base de datos médicos (datos de unas 100 enfermedades diferentes y sus diagnósticos). Como él mismo señala, el programa no nos permite practicar la medicina, pero sí encontrar una base estupenda para nuestra hipocondría. La base de datos muestra que Naylor se está refiriendo a sistemas expertos aplicables al mundo real.

Este es el libro con el que ha de empezar todo aquel que esté interesado en el estudio de los sistemas expertos.

CHESS AND COMPUTERS, David Levy (Potomac, Maryland, Computer Science Science Press, Inc., 1976).

El señor Levy es un maestro internacional del ajedrez y un escritor muy prolífico en este tema. Es famoso mundialmente debido a la ''apuesta Levy''. En 1968 apostó que en diez años no se produciría ningún programa de ajedrez capaz de ganarle. Ganó su apuesta en 1978. Empezó su vida en el campo de la informática en la Universidad de Edinburgo, en donde enseñó programación en ALGOL e Inteligencia Artificial a principios de los años setenta. Levy es una de las autoridades mundiales en el campo del ajedrez en el ordenador y es presidente de una compañía, basada en Londres, que se especializa en la programación de juegos inteligentes.

Quienes estén interesados en la historia del ajedrez, en los diferentes intentos del hombre por conseguir máquinas que jugaran al ajedrez, este es el libro más indicado. Este contiene una historia sólida y ofrece bastantes partidas, o partes de éstas, para mostrar cómo funcionan los programas.

COMPUTER GAMESMANSHIP, David Levy (Londres, Century Publishing, 1983).

Subtitulado como "Guía completa para la creación y estructuración de programas para juegos", este libro, por el sólo hecho de estar escrito por David Levy, ya es recomendable. Contiene una exposición particularmente clara sobre las búsquedas, el algoritmo alfa-beta y las diferentes maneras de acelerar las búsquedas de jugada por parte del ordenador. Entre otros juegos se tocan: el NIM, diversos juegos de cartas (entre los que se incluyen el POKER y el RUMMY), el AJEDREZ, el BACKGAMMON, el OTHELLO, el GO-MOKU, el SHOGI y el DOMINO.

Como se ve, se trata de una lista formidable. Este libro es una necesidad para quienes estén interesados en las relaciones entre inteligencia artificial y la programación de juegos.

COMPUTER GAME-PLAYING, Theory and Practice, M. A. Bramer (Chichester, West Sussex, Ellis Horwood Ltd., 1983).

Aunque de fácil lectura, en éste se trata el tema de un modo mucho más complicado que en el libro de Levy. Sus 300 páginas son muy ricas en ideas específicas aplicables en la creación de programas de juegos, así como sugerencias para otras líneas de investigación. Es una obra de referencia extremadamente valiosa, pese a que no es el tipo de libro que uno se sienta a leer de un tirón (como es el caso del de Levy).

COMPUTER POWER AND HUMAN REASON, Joseph Weizenbaum (San Francisco, W. H. Freeman and Co., 1976).

Es éste un libro importante, aun cuando su tesis fundamental —que con la investigación de la inteligencia artificial estamos rozando, sin ser del todo conscientes de ello, zonas extremadamente peligrosas— reside más en una fijación interna de su autor que en unos argumentos verdaderamente convincentes. Mencionaba este libro en el material que acompañaba al programa DOCTOR, ya que el primer programa ELIZA se lo debemos al autor de este libro. En realidad, es el resultado directo del *shock* sufrido por Weizenbaum al ver la reacción de la gente ante ELIZA.

Weizenbaum cree que hay áreas del conocimiento en las que no se deben introducir las máquinas, aun cuando pudieran hacerlo. Parece que todos los argumentos que se le pudieron ofrecer en contra, como por ejemplo aquél según el cual si una máquina puede ayudar en ciertas circunstancias, se debería investigar, chocaron contra una pared, ya que el tono de Weizenbaum, en el continuo debate que originó la publicación del libro, con el paso de los años se ha ido haciendo cada vez más estridente. McCorduck (en su libro *Machines Who Think*) cuenta la historia de un colega suyo que sostenía que Weizenbaum la había tomado con la inteligencia artificial por la sencilla razón de que no había conseguido ningún programa más de interés desde que produjo ELIZA.

Sea como fuere, el libro merece la pena, aun cuando el lector termine rechazando la mayoría de las tesis que presenta. Siempre está bien el leer una crítica bien informada, aunque en este caso uno puede acabar sintiendo que el crítico está golpeando un tambor vacío. Pero puede que no le produzca la misma impresión a todo el mundo. El hecho de que se sigan haciendo ediciones del libro casi diez años después de su primera edición (algo muy raro en un campo que se desarrolla de un modo tan rápido como la inteligencia artificial), muestra que sus ideas todavía se someten a consideración y debate.

THE FIFTH GENERATION, Artificial Intelligence and Japan's Computer Challenge to the World, Edward A. Feigenbaum y Pamela McCorduck (Reading, Massachusetts, Addison-Wesley Publishing Co., 1983).

Es un libro típicamente americano. Un libro escrito con un objeto determinado: generar una reacción política específica. Como tal, sus autores lo escribieron apresuramente y parece más un artículo largo para una revista como el *Times* que un libro. (En términos de estilo es considerablemente inferior al libro anterior de Pamela McCorduck, *Machines Who Think,* que, pese a la palidez de su título, transmite, bastante poéticamente, una visión particularmente rica de la historia de la inteligencia artificial. Más adelante se ofrece un comentario más detallado sobre este libro.)

Dejando el estilo aparte, el mensaje del libro es importante. Nos viene dado en tres partes: una observación, una advertencia y un llamamiento a las armas. Japón se ha dado cuenta de que la riqueza del mundo del futuro yace en la gestión, manipulación y control de la información. Japón ha decidido que será el líder del mercado en ésta la más moderna de las industrias. Y utilizando el ya conocido poder japonés, ha reunido a un grupo de jóvenes y brillantes cerebros ("nadie por encima de los 35", era una de las reglas) para hacer realidad la inteligencia del ordenador antes de que acabe esta década.

Esta es la observación. La advertencia es que América corre el riesgo de perder para siempre su liderazgo en la tecnología informática. El llamamiento a las armas es para que surja algún tipo de respuesta nacional coordinada (como ya lo han hecho los británicos y otros países europeos) que detenga un poco el reto japonés. Independientemente de que percibamos el riesgo de un modo tan inminente como lo hacen McCorduck y Feigenbaum, e independientemente de que estemos de acuerdo con el remedio propuesto (aun cuando reconozcamos el diagnóstico), este libro debe leerse. Sin las perspectivas que ofrece este libro, tendríamos una visión distorsionada de la inteligencia artificial.

FIFTH GENERATION COMPUTER SYSTEMS, T. Moto-oka (Nueva York, North-Holland Publishing Co., 1982).

Se recopilan en este libro los debates de un congreso internacional sobre sistemas informáticos de la quinta generación que tuvo lugar en Japón en 1981. La quinta generación es una denominación que vamos a oír mucho en los próximos diez años y este libro es el informe de donde se inició todo. Las diferentes ponencias van de las visiones de conjunto a presentaciones detalladas sobre temas muy especializados.

No es un libro para hojear, ya que la lectura no es fácil y sólo vale para aquellos que estén verdaderamente interesados en las implicaciones y promesas del proyecto japonés.

EL LIBRO GIGANTE DE LOS JUEGOS PARA ORDENADOR, Tim Hartnell (Madrid, Anaya Multimedia, 1984).

No creo que haya nada malo en recomendar un libro mío. He incluido esta obra en la bibliografía porque contiene un gran número de programas que demuestran ciertos aspectos de la inteligencia artificial, entre los que se incluye uno que utiliza la "lógica apresurada" para buscar la solución a un código parecido al del "Mastermind" fijado por el usuario. Los programas del REVERSI/OTHELLO, GOMOKU y SHOGUN (mi propia versión del Hasami Shogi) contienen todos ellos códigos que pueden ser interesantes. Hay incluso un juego de AJEDREZ que juega espantosamente mal, pero que puede de-

mostrar el funcionamiento de ciertas funciones de evaluación no especialmente refinadas. Al menos, el lector encontrará en este libro una gran fuente de ideas para utilizar como base de sus propios programas de inteligencia artificial.

INTELLIGENT SYSTEMS, The Unprecedented Opportunity, J. E. Hayes y D. Michie (Chichester, West Sussex, Ellis Horwood Ltd., y Nueva York, Halstead Press, 1983).

Si el libro de Naylor es una guía fundamental a la hora de construirse cada uno sus propios sistemas expertos, este libro es esencial como soporte teórico. De aquí procede la sólida teoría que sostiene el humor de Naylor.

Este libro nació de un seminario patrocinado por la British Computer Society, llamado "Los próximos diez años" y realizado bajo la dirección independiente de Michie. Pese a su forma compuesta de ponencias para el seminario, lo que sugiere que podría ser seco e incomprensiblemente para mortales ordinarios como yo, lo encontré claro y sencillo. Y sobre todo fascinante. Es quizá extraordinario que una parte tan extensa de la inteligencia artificial sea tan apasionante. Nadie que esté interesado en la inteligencia artificial puede encontrar este libro falto de emoción. Con este libro, como en muchos otros, el lector tiene la sensación de estar actuando recíprocamente con una "ciencia ficción" del futuro muy cercana y muy probable.

Y precisamente porque es probable, su fascinación vale más que cien medios imaginarios.

MAQUINAS QUE PIENSAN, Pamela McCorduck (San Francisco, W. H. Freeman and Co., 1979).

Olvidemos el título. Este libro es una verdadera piedra preciosa. Subtitulado "Investigación personal en la historia y el porvenir de la inteligencia artificial", su autora se refiere, con un estilo francamente bueno, a las gentes que componen la historia de la inteligencia artificial.

Empieza el libro haciendo un resumen de todas las tradiciones históricas en las que el hombre se ha asustado de sus invenciones parecidas a él mismo y después pasa a argumentar, de un modo totalmente convincente, que esas actitudes han desvirtuado el modo en que hoy la gente reacciona u opina ante los artefactos que muestran inteligencia artificial.

Este modo de ver el tema es extremadamente valioso en sí mismo y sirve para demostrar que McCorduck sabe muy bien por dónde se anda en este tema. En realidad, habló con todos los gigantes de este campo de la investigación, incluyendo a Weizenbaum, a Simon (quien junto con Newell construyó la primera máquina pensante del mundo, el Logic Theorist, que descubrió una prueba más simple para uno de los teoremas del *Principia Mathematica* que la que habían encontrado Russell y Whitehead; de lo que, al parecer, Russell estaba encantado) y Rosen (quien, con su equipo del Stanford Research Institute, construyeron a Shakey, un robot que se movía solo y que en los años sesenta se hizo muy famoso).

Todo esto significa que el libro es una importante fuente de documentos históricos, llena de apartes y anécdotas que muestran el lado humano de la investigación de la inteligencia artificial (así como sus hitos y sus fracasos).

PRACTICAL EXPERIENCE OF MACHINE TRANSLATION, Veronica Lawson (Nueva York, North Holland Publishing Co., 1982).

Este libro es la recopilación de los debates de un congreso que tuvo lugar en Londres, en el mes de noviembre de 1981, y en el que se reunieron un gran número de expertos en el campo de la traducción mediante máquinas. Es un informe excelente del estado de este campo en aquel momento y marca asimismo las líneas fundamentales en la investigación de la traducción con ordenadores.

Entre los títulos de las ponencias presentadas se pueden citar: ''La traducción mediante máquinas y la gente'' (Veronica Lawson); ''Aspectos económicos de la traducción mediante máquinas'' (Georges van Slype); ''Experiencia en la elaboración posterior de textos traducidos del inglés al francés'' (Bernard Lavorel), y ''Los límites de la innovación en la traducción mediante ordenadores'' (Margaret Masterman).

Este libro es un buen empiece si se está interesado en este campo específico de la inteligencia artificial.

THOSE AMAZING ELECTRONIC THINKING MACHINES!, Isaac Asimov, Martin S. Greesberg y Charles E. Waugh (Nueva York, Frankling Watts, 1983).

Esta antología, recopilada por Asimov, de historias de robots y ordenadores, se encuentra aquí incluida porque es fascinante ver las posibilidades que los escritores han podido llegar a imaginar para este arte.

Cuando el lector esté cansado de símbolos lógicos, puede darse un respiro leyendo algún relato de esta antología.

3
Glosario de términos informáticos y de inteligencia artificial

ACUMULADOR: Parte de la unidad aritmético-lógica del ordenador que almacena los resultados intermedios.

ALFANUMERICO: Se utiliza generalmente para describir el teclado que contiene teclas alfabéticas y numéricas, en comparación con el teclado numérico, que sólo tiene teclas para los números del uno al nueve.

ALGEBRA BOOLEANA: El álgebra del mecanismo lógico y de toma de decisiones desarrollada por el matemático inglés George Boole. De ella depende la capacidad que muestra el ordenador a la hora de tomar decisiones. Asimismo, gran parte de la fuerza del ordenador para demostrar un comportamiento inteligente reside en el uso del Algebra Booleana (véase el capítulo dedicado al APRENDIZAJE Y RAZONAMIENTO).

ALGORITMO: La secuencia de pasos dados en la solución de un problema.

APL: Siglas de *Automatic Programming Language,* un lenguaje desarrollado por Iverson, a principios de los años sesenta, que apoya una amplia serie de operadores y estructuras de datos. Usa un conjunto de caracteres no estándar.

ARCHIVO *(File):* Grupo de datos relacionados entre sí que se procesan juntos, tales como el archivo de un estudiante, etc.

ASCII: Siglas de *American Standar Code for Information Exchange.* Se trata de un código casi universal para caracteres, números y símbolos, asignándole un número entre el 0 y el 255 a cada uno de ellos, como por ejemplo, el 65 para la letra A.

BASE DE CONOCIMIENTO: En los sistemas expertos se llama así a la base de datos sobre los cuales el sistema emite juicios.

BASE DE DATOS: Colección de datos organizada de tal modo que el ordenador pueda

acceder rápidamente a ella. Una base de datos relacional es aquella en la que las conexiones entre los distintos elementos que forman la base de datos están almacenadas explícitamente con el fin de ayudar a la manipulación y el acceso a éstos.

BANDERA: Indicador interior del programa que da información sobre una condición determinada.

BASIC: Siglas de *Beginner's All-purpose Symbolic Instruction Code;* es el lenguaje que se utiliza generalmente en los microordenadores. Es fácil de aprender, ya que la mayoría de las instrucciones son muy parecidas al inglés.

BAUDIO: Unidad que mide la velocidad de transmisión de los datos. Normalmente representa el número de bits (unidades discretas de información) por segundo.

BENCHMARK (véase "Punto de referencia").

BINARIO: Sistema de numeración en el que hay sólo dos símbolos, 0 y 1 (en oposición al sistema decimal ordinario, en el que hay diez símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9). Los ordenadores "piensan" en términos binarios.

BOOTSTRAP: Primer programa que se ejecuta tras la conexión del ordenador y que permite a éste aceptar y comprender otros programas.

BUFFER (véase "Regulador").

BUG: Error en un programa.

BUS: Grupo de conexiones eléctricas usadas para unir un ordenador a otro mecanismo auxiliar o a otro ordenador.

BYTE: El grupo más pequeño de bits que forma una palabra en el ordenador. Generalmente un ordenador se describe por sus bits, ya sea de ocho bits o de dieciséis bits, lo que significa que la palabra consiste en una combinación de ocho o de dieciséis ceros o unos.

CAMPO: Colección de caracteres que forman un grupo distinto, como un código de identificación, un nombre o una fecha; generalmente un campo forma parte de una información.

CIRCUITO PRINCIPAL *(Motherboard):* Unidad equipada de modo que se le pueden añadir al ordenador circuitos adicionales proporcionándole así características de las que carece ese modelo estándar (tales como gráficos de alta resolución o control de robot).

COBOL: Siglas de *Common Business Orientated Language,* un lenguaje de programación estándar, muy parecido al inglés, que se utiliza fundamentalmente en el mundo de los negocios.

CODIGO DE CARACTERES: Es el número de ASCII (véase ASCII) que se refiere a un símbolo en particular, tales como 32 para un espacio o 65 para la letra A.

COMPARTIDO: Se utiliza este término cuando se da el caso de un gran número de usuarios, cada uno en un terminal distinto, que utilizan el mismo ordenador, el cual divide su tiempo entre ellos de modo que da la impresión de que les está dedicando "toda su atención" a cada uno por separado.

COMPILADOR: Programa que pasa otro programa escrito en un lenguaje de alto nivel (parecido al humano) al lenguaje de la máquina de modo que ésta lo entienda perfectamente.

CONCATENAR: Se llama concatenación a la suma de dos series.

CP/M: Siglas de *Control Program/Microcomputer;* se trata de un sistema universal de

operación por medio de disco, desarrollado y puesto en el mercado por Digital Research, en Pacific Grove, California.

DATOS: Término general para la información procesada por un ordenador.
DEBUG: Quitar los errores *(Bugs)* de un programa.
DEDUCCION SIMBOLICA (véase "Sistemas deductivos").
DIAGRAMA DE FLUJO (*Flowchart):* Trazado escrito de la escritura y curso de un programa en el que se utilizan formas diferentes, como un rectángulo o un cuadrado para indicar una acción del ordenador, y un rombo para las decisiones tomadas por éste. Normalmente, se suele hacer el gráfico o diagrama del programa antes de introducir una sola línea de éste en el ordenador.
DIRECCION: Número que se refiere a la posición, generalmente en la memoria del ordenador, en la que se almacena la información.
DISCO: Un medio de almacenaje magnético (más adelante descrito también como "disco duro" o "disco flexible") utilizado para archivar la información del ordenador y los programas. Estos discos se parecen hasta cierto punto a los discos de música de 45 rpm y miden generalmente ocho, cinco y un cuarto, o tres pulgadas de diámetro. En algunos sistemas también existen "microdiscos" más pequeños todavía.
DISCO FLEXIBLE (váse "Disco").
DOBLE DENSIDAD: Adjetivo con el que se describe a los discos grabados utilizando una técnica especial que, como su nombre sugiere, duplica la cantidad de almacenaje que el disco proporciona.
DOCUMENTACION: Las instrucciones y aplicaciones escritas que acompañan al programa.
DOS: Siglas de *Disck Operating System:* el programa que permite que el ordenador controle un sistema de disco.

ENSAMBLADOR *(Assembler):* Es un programa que convierte otro programa escrito en lenguaje de ensamble *(assembly language)* en un código que el microprocesador puede ejecutar directamente.
ENTRADA *(Input):* Toda la información que se introduce en un programa durante su ejecución.
EPROM: Siglas de *Erasable Programmable Read Only Memory,* un mecanismo que contiene información bajo una forma semipermanente y que exige una exposición a la luz ultravioleta en el caso de querer borrar su contenidos.

FIRMWARE: Los componentes físicos, la maquinaria de un ordenador, se denominan *hardware;* los programas, en forma que la máquina pueda leerlos, ya sean en disco o en cassettes, se llaman *software,* y los programas que funcionan dentro del ordenador relacionados íntimamente con el *hardware* se denominan *firmware;* el *firmware* puede alterarse, hasta cierto punto, por medio del *software.*
FORTRAN: Un lenguaje de alto nivel que se utiliza generalmente para el trabajo científico (siglas de *FORmula TRANslation).*

GATE ("Puerta lógica): "Componente" del ordenador que se encarga de tomar decisiones permitiendo para ello que el circuito vaya en una u otra dirección según las

condiciones que se han de satisfacer (véase el capítulo dedicado al APRENDIZAJE Y RAZONAMIENTO).

GIGO: Siglas de *Garbage In Garbage Out,* con lo que se quiere indicar que si se introduce basura *(garbage),* es decir, datos erróneos, el resultado de tales datos será también "basura".

GLOBAL: El conjunto de condiciones que afectan a todo un programa se denomina "global", opuesto a "local" en el caso de que no afecte a todo el programa.

GRAFICOS: Término con que se designa toda salida del ordenador que no sea alfanumérica o simbólica.

HARD COPY: La información que la impresora vierte en papel.

HARDWARE: Se denomina *hardware* a la maquinaria física del ordenador (véase *firmware* y *software*).

HEURISTICA: Manera de alcanzar un objetivo basada en la experiencia más que en el cálculo; esta técnica no garantiza un resultado cierto (comparada a un algoritmo que es un procedimiento que, cuando se aplica, produce siempre el resultado deseado); los programas de ajedrez juegan, en gran medida, de manera heurística.

HEXADECIMAL: Un sistema de contar muy apreciado por los programadores de códigos, ya que está en estrecha relación con el método de almacenaje utilizado por los ordenadores; está basado en el número 16 en vez del 10, que es el número en el que se basa nuestro sistema de contar ordinario.

IMPRESORA DE MATRIZ DE PUNTOS *(Dot-Matrix Printer):* Impresora que forma las letras y símbolos con puntitos, normalmente en una cuadrícula de ocho por ocho o de siete por cinco.

IMPRESORA TERMICA: Mecanismo que imprime la salida *(Output)* del ordenador en un papel sensible al calor. Aunque los impresores térmicos son más silenciosos que los otros, su salida no siempre es fácil de leer y además el papel no es fácil de almacenar.

INGENIERIA DEL CONOCIMIENTO: Proceso mediante el cual se traspasa al ordenador la pericia o conocimiento humano en un campo determinado, o varios.

INSTRUCCION: Elemento del código de programación que le dice al ordenador que lleve a cabo una tarea determinada. Una instrucción en lenguaje de ensamble, por ejemplo, podría ser ADD, con lo que le dice al ordenador que realice una suma.

INTELIGENCIA ARTIFICIAL: Dominio de la ciencia informática que se centra en provocar en la máquina un comportamiento que si proviniera de un ser humano se diría que es inteligente. Uno de los objetivos de la inteligencia artificial es hacer que los ordenadores sean más útiles para todo el mundo. La investigación de la inteligencia artificial puede resultar también muy útil a la hora de ayudarnos a comprender nuestros propios procesos mentales.

INTERPRETE: Convierte el programa escrito en un lenguaje de alto nivel (comprensible para el humano) en un programa comprensible para el ordenador.

KILOBYTE: Unidad de medida del lenguaje; un kilobyte (generalmente abreviado en K) equivale a 1.024 bits.

LENGUAJE DE ALTO NIVEL: Lenguaje de programación parecidos a la lengua del usuario (generalmente al inglés). Los lenguajes de bajo nivel están más cercanos a los lenguajes comprensibles para el ordenador. Como los lenguajes de alto nivel han de pasarse a una forma que el ordenador pueda comprender antes de procesarlos, su ejecución es más lenta que la de los lenguajes de bajo nivel.

LENGUAJE DE BAJO NIVEL: Lenguaje parecido al utilizado en el ordenador.

LENGUAJE MAQUINA: Es el escalón inferior al lenguaje de bajo nivel; es el lenguaje que el ordenador entiende directamente.

MEMORIA: El mecanismo o mecanismos utilizados por el ordenador para guardar la información y los programas que se procesan, así como el conjunto de instrucciones que le indican a éste el modo de llevar a cabo el programa. Básicamente existen dos tipos de memoria (véase RAM y ROM).

MEMORIA DINAMICA: Memoria que ha de recargarse continuamente con el fin de mantener sus contenidos.

MEMORIA VOLATIL: Un mecanismo de memoria que pierde sus contenidos cuando se le corta la corriente.

MICROPROCESADOR: El *chip* que hace de corazón del ordenador o de su parte pensante.

MODEM: Siglas de *MOdulator/DEModulator,* mecanismo que permite que el ordenador se comunique con otro por medio del teléfono.

MONITOR: *a*) una pantalla de televisión utilizada como unidad de *display;* carece de mandos de ajuste; *b*) la información que lleva el ordenador y que le permite comprender y ejecutar las instrucciones del programa.

MOUSE: Unidad de control, ligeramente más pequeña que un paquete de tabaco, que rueda sobre la mesa y mueve en sentido paralelo un cursor sobre la pantalla con el fin de seleccionar las diferentes opciones dentro del programa. Los *Mouses* funcionan, ya sea por la acción de sus ruedas o leyendo una cuadrícula en la superficie sobre la que se mueven.

OCTAL: Sistema de numeración basado en el ocho (utilizando los números 0, 1, 2, 3, 4, 5, 6 y 7).

PALANCA DE MANDO: Generalmente utilizada en los programas de juegos, se trata de un mecanismo que introduce una señal en el ordenador; una señal que está relacionada con la posición que ocupa la palanca.

PASCAL: Un lenguaje de alto nivel desarrollado a finales de los años sesenta por Niklaus Wirth y que anima a una programación disciplinada y estructurada.

PILOT: Lenguaje de alto nivel que se suele utilizar para desarrollar los programas utilizados en la educación.

PONER A CERO *(Reset):* Señal que vuelve al ordenador al punto en el que estaba al encenderlo.

PORT: Punto de entrada o salida mediante el cual se transfieren los datos.

PROCESADOR DE TEXTOS *(Word Processor):* Un ordenador que da acceso a una "máquina de escribir inteligente"con una amplia gama de posibilidades de ajuste y corrección (o un ordenador que opere un programa de proceso de textos).

PROGRAMA: Serie de instrucciones que sigue el ordenador para llevar a cabo una tarea determinada.

PUNTO DE REFERENCIA *(Benchmark):* Prueba que se realiza para medir algún aspecto del funcionamiento del ordenador comparándola al resultado de ejecutar una prueba similar en otro ordenador.

RAM: Siglas de *Random Access Memory;* es la memoria que permite al ordenador mantener el programa en curso de ejecución. Los contenidos del RAM se pueden cambiar, mientras que los del ROM no pueden cambiarse bajo el control del *software.*

RED *(Network):* Grupo de ordenadores que funcionan juntos.

REGULADOR *(Buffer):* Mecanismo de almacenaje que contiene la entrada mediante una tecla o similar, soltándola posteriormente según la frecuencia que le vaya dictando el ordenador.

REFRESCO: Las memorias dinámicas han de recibir irrupciones periódicas de electricidad con el fin de mantener sus contenidos. La señal que recuerda a la memoria cuáles son sus contenidos se llama señal de refresco.

REGISTRO: Punto de la memoria del ordenador que contiene datos de uso frecuente.

REPRESENTACIÓN: La particular organización de la información dentro del ordenador de modo que pueda ser manipulada por los sistemas de control de la base de datos o conocimiento.

ROM (véase RAM).

RS-232: Norma de conexión estándar que conecta un Modem, o el equipo asociado a la terminal, a un ordenador.

S-100 BUS: Se trata también de una norma de conexión estándar (véase RS-232) hecha con 100 líneas de comunicación común paralelas que se utilizan para conectar los circuitos dentro de los microordenadores.

SALIDA *(Output):* Todos los datos producidos por el ordenador mientras está procesando, ya aparezcan estos datos en la pantalla, ya los vierta a la impresora o los utilice internamente.

SNOBOL: Lenguaje de alto nivel, desarrollado por los laboratorios Bell, que utiliza el reconocimiento de esquemas o dibujos y la manipulación de las cadenas.

SEÑALIZADOR *(Flag)* (véase "Bandera").

SISTEMA DE PROCESADO DE INFORMACION Y CONOCIMIENTOS: Se trata de los sistemas de ordenadores llamados de la Quinta Generación que se están desarrollando actualmente en Japón (véanse las referencias a los libros *The Fifth Generation* y *Fifth Generation Computer Sistems* que aparecen en el apéndice de bibliografía).

SISTEMA DEDUCTIVO: El mecanismo mediante el cual un programa llega a conclusiones (véase el capítulo dedicado al APRENDIZAJE Y RAZONAMIENTO); algunos programas toman decisiones raídas y tajantes, otros operan en el mundo de la lógica complicada en la que se permite cierto grado de incertidumbre.

SISTEMA "LLAVE EN MANO": Generalmente utilizado en el mundo de los negocios, este sistema está preparado para ejecutar directamente y sólo hay que pulsar una tecla para que funcione.

SOFTWARE ("logical"): Es el programa que ejecuta el ordenador.

SOFTWARE APLICADO: Programas escritos para la realización de tareas especiales, como el procesado de palabras o listas de correspondencia.

SOFTWARE DE SISTEMAS: Secciones de códigos que llevan a cabo tareas administrativas dentro del ordenador o ayudan en la escritura de otros programas, pero que no se usan para realizar la tarea que se quiere que ejecute el ordenador.

SINTAXIS: Como en las lenguas humanas, la sintaxis es el conjunto de reglas estructurales que gobiernan el uso del lenguaje en el ordenador.

SUBRUTINA: Parte del código, o programa, a la que se llama un número determinado de veces durante la ejecución del programa.

TECLADO HEX *(Hex Pad):* Teclado, parecido al de una calculadora, que se usa para introducir directamente los números hexadecimales.

TECLADO NUMERICO *(Numeric Pad):* Mecanismo utilizado para introducir información numérica en el ordenador, parecido al teclado de una calculadora.

TIEMPO REAL: Cuando una acción realizada en el ordenador progresa paralelamente al tiempo del ''mundo real'', se dice que la acción ocurre en tiempo real. Un ejemplo sería un programa que mostrara el desarrollo de una colonia de bacterias que se reprodujeran con el mismo ritmo de crecimiento con que lo haría una colonia real. Ultimamente han aparecido muchos juegos que requieren reacciones en tiempo real. La mayoría de las máquinas de los salones recreativos transcurren en tiempo real.

UNIDAD CENTRAL DE PROCESADO: Es el corazón del ordenador, en donde se llevan a cabo las funciones aritméticas, lógicas y de control.

VLSI: Siglas que significan ''integración de componentes a gran escala'' en un *chip.* Se está desarrollando esta técnica en Japón y Estados Unidos y su objetivo es conseguir *chips* con el equivalente a 10 millones de transistores (los *chips* fabricados actualmente contienen el equivalente a medio millón de transistores como máximo).

4
Programas para el ZX Spectrum

TICTAC

```
  10 REM ** TIC TAC **
  15 INK 7: PAPER 1: FLASH 0: BR
IGHT 0: BORDER 1
  20 GO SUB 1180: REM INICIALIZA
CION
  30 REM ** REQUISITOS  PREVIOS
AL JUEGO **
  40 FOR J=1 TO 9
  50 LET A(J)=32
  60 NEXT J
  70 FOR J=1 TO 5
  80 LET D(J)=0
  90 NEXT J
 100 LET COUNT=0
 110 LET R$=""
 120 GO SUB 1070: REM TABLERO
 130 REM *** CICLO PRINCIPAL ***
 140 GO SUB 540: REM MOVIMIENTO
DE LA MAQUINA
 150 GO SUB 1070: REM TABLERO
 160 GO SUB 870: REM COMPRUEBA L
A VICTORIA
 170 IF R$<>"" THEN  GO TO 240
 180 GO SUB 980: REM ACEPTAR MOV
IMIENTO HUMANO
 190 GO SUB 1070: REM TABLERO
 200 GO SUB 870: REM COMPRUEBA L
A VICTORIA
 210 IF R$="W" THEN  GO TO 140
 220 REM **FIN CICLO PRINCIPAL**
 230 REM ***********************
 240 REM FIN DEL JUEGO
 250 GO SUB 1070: REM TABLERO
 260 PRINT : PRINT
 270 IF R$="W" THEN  PRINT TAB 8
;"YO GANO": FOR G=1 TO 4: BEEP .
2,5: BEEP .1,10: BEEP .4,9:: NEX
T G: LET FLAG=-1: GO TO 1370
 280 IF R$="L" THEN  PRINT TAB 8
;"TU GANAS": FOR G=1 TO 16 STEP
2: BEEP .1,G: BEEP .1,G+2: NEXT
G: LET FLAG=1: GO TO 1370
 290 IF R$="D" THEN  PRINT TAB 8
;"TABLAS": FOR G=1 TO 4: BEEP .2
,5: BEEP .1,G: BEEP .4,9: NEXT G
: LET FLAG=0: GO TO 1370
 300 REM ACTUALIZACION DE LA BAS
E DE DATOS
 310 FOR B=1 TO 5
 320 FOR J=2 TO 9
 330 IF M(J)=D(B) THEN  GO SUB 3
70
 340 NEXT J
 350 NEXT B
 360 GO TO 430
```

```
 370 REM REORDENACION DE LOS ELE
MENTOS DE LA MATRIZ M
 380 LET TEMP=M(J+FLAG)
 390 LET M(J+FLAG)=M(J)
 400 LET M(J)=TEMP
 410 LET J=9
 420 RETURN
 430 PRINT : PRINT
 440 PRINT "ESTA ES MI PRIORIDAD
 ACTUALIZADA"
 450 PRINT : PRINT
 460 FOR J=1 TO 9
 470 PRINT M(J);" ";
 480 NEXT J
 490 PRINT : PRINT
 500 PRINT "PULSE ""ENTER"" PARA
 CONTINUAR"
 510 INPUT A$
 520 GO TO 120
 540 REM MOVIMIENTO DE LA MAQUIN
A
 550 LET P=CODE "O"
 560 LET X=0
 570 LET J=1
 580 IF A(W(J))=A(W(J+1)) AND A(
W(J+2))=32 AND A(W(J))=P THEN  L
ET X=W(J+2): GO TO 750
 590 IF A(W(J))=A(W(J+2)) AND A(
W(J+1))=32 AND A(W(J))=P THEN  L
ET X=W(J+1): GO TO 750
 600 IF A(W(J+1))=A(W(J+2)) AND
A(W(J))=32 AND A(W(J+1))=P THEN
 LET X=W(J): GO TO 750
 610 IF J<21 THEN  LET J=J+3: GO
 TO 580
 620 IF P=CODE "O" THEN  LET P=C
ODE "X": GO TO 570
 630 REM ** SI NO GANA SE BUSCA
UN MOVIMIENTO DE BLOQUEO **
 640 REM * ENTONCES SE USA LA SI
GUIENTE SECCION *
 650 LET J=1
 660 IF A(M(J))=32 THEN  LET X=M
(J): GO TO 750
 670 IF J<10 THEN  LET J=J+1: GO
 TO 660
 680 LET H=0
 690 LET H=H+1
 700 LET X=INT (RND*9)+1: IF A(X
)=32 THEN  GO TO 750
 710 IF H<100 THEN  GO TO 690
 720 LET R$="D": REM ES UN DIBUJ
O
 750 REM HACER MOVIMIENTO
 760 LET A(X)=CODE "O"
 770 LET COUNT=COUNT+1
 780 LET D(COUNT)=X
 790 LET FLAG=0
 800 FOR J=1 TO 9
 810 IF A(J)=32 THEN  LET FLAG=1
 820 NEXT J
 830 IF FLAG=0 AND R$="" THEN  L
ET R$="D"
 840 REM SI TODAS LAS POSICIONES
 ESTAN LLENAS, Y NO ESTA ASIGNAD
O R$, ES UN DIBUJO
 850 RETURN
 860 REM *****************
 870 REM COMPRUEBA LA VICTORIA
 880 LET J=1
 890 IF A(W(J))=32 THEN  LET J=J
+3
 900 IF J>23 THEN  RETURN
 910 IF A(W(J))=A(W(J+1)) AND A(
W(J))=A(W(J+2)) THEN  GO TO 940
 920 IF J<22 THEN  LET J=J+3: GO
 TO 890
 930 RETURN
 940 IF A(W(J))=CODE "O" THEN  L
ET R$="W": REM GANA LA MAQUINA
 950 IF A(W(J))=CODE "X" THEN  L
ET R$="L": REM PIERDE LA MAQUINA

 960 RETURN
 970 REM ****************
 980 REM MOVIMIENTO HUMANO
 990 PRINT : PRINT
1000 PRINT "Entre su movimiento.
"
1010 INPUT MOVE
1020 IF MOVE<1 OR MOVE>9 THEN  G
O TO 1010
1030 IF A(MOVE)<>32 THEN  GO TO
1010
1040 LET A(MOVE)=CODE "X"
1050 RETURN
1060 REM *********************
1070 REM TABLERO
1080 CLS
1090 PRINT : PRINT : PRINT
1100 PRINT "1 : 2 : 3   ";CHR$ A
(1);" : ";CHR$ A(2);" : ";CHR$ A
(3)
1110 PRINT "---------   --------
-"
1120 PRINT "4 : 5 : 6   ";CHR$ A
(4);" : ";CHR$ A(5);" : ";CHR$ A
(6)
1130 PRINT "---------   --------
-"
1140 PRINT "7 : 8 : 9   ";CHR$ A
(7);" : ";CHR$ A(8);" : ";CHR$ A
(9)
1150 PRINT
1160 RETURN
1170 REM *********************
```

```
1180 REM INICIALIZACION
1190 CLS
1200 DIM A(9): REM TABLERO
1210 DIM M(10): REM ACCESO A LA
BASE DE DATOS
1220 DIM W(24): REM DATOS DE VIC
TORIA O BLOQUEO
1230 DIM D(5): REM ACCESO AL MOV
IMIENTO EN EL JUEGO ACTUAL
1240 REM DATOS DE VICTORIA O BLO
QUEO
1250 FOR J=1 TO 24
1260 READ W(J)
1270 NEXT J
1280 DATA 1,2,3,4,5,6,7,8,9
1290 DATA 1,4,7,2,5,8,3,6,9
1300 DATA 1,5,9,3,5,7
1310 REM BASE INICIAL DE DATOS
1320 FOR J=1 TO 10
1330 READ M(J)
1340 NEXT J
1350 DATA 2,6,8,4,7,3,1,9,5,2
1360 RETURN
1370 PRINT : PRINT
1380 PRINT "SI QUIERES JUGAR OTR
A VEZ"
1390 PRINT "ENTRA ""S"" Y PULSA
""ENTER"""
1395 PRINT : PRINT
1400 PRINT "SI NO, PULSA SOLO ""
ENTER"""
1410 INPUT R$
1420 IF R$="S" THEN  GO TO 30
1430 CLS
1500 STOP
4000 REM ***********************
********************************
4050 REM ** PARA JUGAR CONTRA UN
 OPONENTE QUE ACTUE DE MANERA AL
EATORIA
4100 REM SE DEBE CAMBIAR LA LINE
A 140 POR "GOSUB 4500"
4200 REM DE ESTE MODO SE ENTRA E
N LA SIGUIENTE SUBRUTINA
4500 REM MOVIMIENTO ALEATORIO DE
 UNA PERSONA
4510 LET H=0
4520 LET H=H+1
4530 LET MOVE=INT (RND*9+1)
4540 IF A(MOVE)=32 THEN  LET A(M
OVE)=CODE "O": RETURN
4550 IF H<100 THEN  GO TO 4520
4560 LET R$="D"
4570 RETURN
```

SILOGISMOS

```
  10 REM *** SILOGISMOS ***
  15 POKE 23658,200: FLASH 0: BR
IGHT 0: CLS : PAPER 1: BORDER 1:
 INK 7
  20 GO SUB 1050: REM INICIALIZA
CION
  30 PRINT : INPUT A$
  35 LET Q$=A$
  40 IF A$="?" THEN  GO TO 910
  50  IF LEN A$<2 THEN  STOP
  60  LET FLAG=0
  70 REM ADVIERTA QUE HAY UN ESP
ACIO EN BLANCO ANTES DE CERRAR C
OMILLAS EN LAS PROXIMAS 5 LINEAS
  80 IF A$( TO 3)="ES " THEN  GO
 TO 480
  85 IF A$( TO 3)="EL " THEN  LE
T A$=A$(4 TO )
  90 IF A$( TO 3)="LA " THEN  LE
T A$=A$(4 TO )
 100 IF A$( TO 3)="UN " THEN  LE
T A$=A$(4 TO )
 110 IF A$( TO 4)="UNA " THEN  L
ET A$=A$(5 TO )
 120 LET X=LEN A$
 130 LET N=0
 140 LET N=N+1
 150 IF A$(N)=" " THEN  LET B$=A
$( TO N-1): GO TO 180: REM EXTRA
E EL PRIMER NOMBRE
 160 IF N<X THEN  GO TO 140
 170 INK 5: PRINT "NO COMPRENDO"
: INK 7: GO TO 30
 180 LET K=4
 190 IF A$(N+1 TO N+2)="F" THEN
 LET K=5
 200 LET C$=A$(N+K TO ): REM PRE
PARATIVOS DE LA FRASE
 210 REM ELIMINACION DE ARTICULO
S
 220 IF C$( TO 4)="UNA " THEN  L
ET C$=C$(5 TO )
 225 IF C$( TO 3)="UN " THEN  LE
T C$=C$(4 TO )
 230 IF C$( TO 3)="EL " THEN  LE
T C$=C$(4 TO )
 235 IF C$( TO 3)="LA " THEN  LE
T C$=C$(4 TO )
 240 REM *ALMACENA INFORMACION*
 250 REM ** PRIMERO SE COMPRUEBA
 SI SE PUEDE ENCONTRAR UN SUJETO
ANTES DE UN ESPACIO EN BLANCO **
 260 LET N=0
 270 LET N=N+1
 280 IF Z$(1,N, TO LEN B$)=B$ TH
EN  GO TO 320: REM EXISTE UN SUJ
ETO AL PRINCIPIO DE LA FRASE
 290 IF Z$(1,N, TO 5)="     " TH
EN  LET Z$(1,N)=B$: GO TO 320
 300 IF N<25 THEN  GO TO 270
 310 INK 2: PRINT "NO ME QUEDA E
SPACIO PARA ALMACENAR MAS SUJETO
S": INK 7
 320 REM ** EL PROGRAMA HA LLEGA
DO AQUI CON EL SUJETO ENCABEZAND
O EL ALMACENAMIENTO **
 330 REM ** AHORA SE PONE EL OBJ
ETO DIRECTO DEBAJO DE EL **
 340 LET K=0
 350 LET K=K+1
 360 IF Z$(K,N, TO LEN C$)=C$ TH
EN  GO TO 400: REM LA INFORMACIO
N YA HA SIDO ALMACENADA DEBAJO D
EL ENCABEZAMIENTO
 370 IF Z$(K,N, TO 5)="     " TH
EN  LET Z$(K,N)=C$: GO TO 400
 380 IF K<25 THEN  GO TO 350
 390 INK 5: PRINT "NO TENGO MAS
ESPACIO PARA ALMACENAR OBJETOS D
IRECTOS.": INK 7
 400 IF FLAG=1 THEN  PRINT Q$: I
NK 5: PRINT TAB 6;">DE ACUERDO<"
: INK 7: GO TO 30: REM SE HA REA
LIZADO EL INTERCAMBIO
 410 REM ** AHORA SE INTERCAMBIA
N EL OBJETO Y EL SUJETO Y SE ALM
ACENA DE NUEVO **
 420 LET FLAG=1
 430 LET M$=B$
 440 LET B$=C$
 450 LET C$=M$
 460 GO TO 250
 470 REM *****************
 480 REM ** CONCLUSIONES **
 490 REM ** PRIMERAMENTE SE FRAG
MENTA LA FRASE **
 500 LET H$=A$: LET A$=A$(4 TO )
: REM SE IGNORA "ES"
 510 IF A$( TO 4)="UNA " THEN  L
ET A$=A$(5 TO ): REM SE IGNORA "
UNA"
 520 IF A$( TO 3)="EL " OR A$( T
O 3)="UN " OR A$( TO 3)="LA " TH
EN  LET A$=A$(4 TO ): REM SE IGN
ORA "EL","UN" O "LA"
 530 REM ** SE COGE LA PRIMERA P
ALABRA - F$ **
 540 LET X=LEN A$
```

```
 550 LET N=0
 560 LET N=N+1
 570 IF A$(N)=" " THEN  LET F$=A
$( TO N-1): GO TO 600
 580 IF N<X THEN  GO TO 560
 590 PRINT TAB 6; INK 5;"NO LO C
OMPRENDO": GO TO 30
 600 REM ** AHORA SE COGE LA SEG
UNDA PALABRA - S$ **
 610 LET S$=A$(N+4 TO )
 620 IF LEN S$<=0 THEN  GO TO 63
0
 621 IF S$(1)=" " THEN  LET S$=S
$(2 TO ): REM IGNORA OTRO ESPACI
O SI ES "UNA"
 630 PRINT H$'TAB 9; INK 5;"(EST
OY BUSCANDO '";F$;"')"
 640 LET X=0
 650 LET X=X+1
 660 IF Z$(1,X, TO LEN F$)=F$ TH
EN  PRINT TAB 10; INK 5;"(ENCONT
RADO EN 1,";X;")": GO TO 700
 670 IF X<25 THEN  GO TO 650
 680 PRINT TAB 6; INK 5;"NO PUED
O ENCONTRAR EL SUJETO": PRINT TA
B 8;F$
 690 GO TO 30
 700 LET Y=1
 710 LET Y=Y+1
 720 IF Z$(Y,X)=S$ THEN  INK 5:
PRINT TAB 6;"> SI": INK 7:: GO T
O 30
 730 IF Y<25 THEN  GO TO 710
 740 LET Y=1
 750 LET Y=Y+1
 760 LET P$=Z$(Y,X)
 770 LET M=0
 780 LET M=M+1
 790 IF Z$(1,M)=P$ THEN  GO TO 8
30
 800 IF M<25 THEN  GO TO 780
 810 IF Y<25 THEN  GO TO 750
 820 PRINT TAB 6; INK 5;"> NO":
GO TO 30
 830 LET Q=1
 840 LET Q=Q+1
 850 IF Z$(Q,M, TO LEN S$)=S$ TH
EN  PRINT TAB 6; INK 5;"> SI": G
O TO 30
 860 IF Q<25 THEN  GO TO 840
 870 IF M<25 THEN  GO TO 780
 880 GO TO 820
 900 REM ***********************
 910 REM COMPRUEBA EL CONTENIDO
        DE UN ARCHIVO PARTICULAR
 920 INK 5: INPUT "QUE SUJETO SE
 VA A COMPROBAR?   ";H$
 930 INK 7: LET T=0
 940 LET T=T+1
 950 IF Z$(1,T, TO LEN H$)=H$ TH
EN  GO TO 990
 960 IF T<25 THEN  GO TO 940
 970 PRINT  INK 5;"NO TENGO DATO
S ALMACENADOS EN ";H$
 980 GO TO 30
 990 LET K=1
1000 LET K=K+1
1010 IF Z$(K,T, TO 5)<>"     " T
HEN  PRINT  INK 5;K;" ";T;" ";Z$
(K,T)
1020 IF K<25 THEN  GO TO 1000
1030 GO TO 30
1040 REM *******************
1050 REM ** INICIALIZACION **
1060 CLS
1080 DIM Z$(25,25,27)
1090 RETURN
```

RISITAS

```
  10  REM ** RISITAS **
  15 POKE 23658,200: BRIGHT 0: F
LASH 0: INK 7: PAPER 1: BORDER 1
  20 GO SUB 2070: REM INICIALIZA
CION
  30 GO SUB 1760: REM PINTAR TAB
LERO
  40 REM ** COMIENZO CICLO PRINC
IPAL **
  50 GO SUB 190: REM MOVIMIENTOS
 DE LA MAQUINA
  60 GO SUB 1760: REM PINTAR TAB
LERO
  70 IF CS>4 THEN  GO TO 120
  80 GO SUB 1950: REM ACEPTAR MO
VIMIENTO HUMANO
  90 GO SUB 1760: REM PINTAR TAB
LERO
 100 IF HS<=5 THEN  GO TO 50
 110 REM *******************
 120 REM      FIN DEL JUEGO
 130 PRINT : PRINT "FIN DEL JUEG
O"
 140 PRINT
 150 IF HS>CS THEN  FOR I=1 TO 2
2 STEP 2: BEEP .1,I: BEEP .1,I+2
: BEEP .2,4: NEXT I: PRINT  FLAS
H 1; INK 1; PAPER 7;"  >> ME HAS
 GANADO <<  "
 160 IF CS>HS THEN  FOR I=20 TO
5 STEP -1.5: BEEP .2,I: BEEP .3,
I+2: NEXT I: FOR I=1 TO 4: BEEP
.1,3: BEEP .2,-1: NEXT I: BEEP 2
,20: PRINT  FLASH 1; INK 2; PAPE
R 0;"   ***  YO SOY EL GANADOR
***"
 170 STOP
 180 REM **********************
 190 REM   MOVIMIENTO  MAQUINA
 200 REM **********************
 210 REM BUSCAR CAPTURAS
 220 LET GSAFE=0: REM PARA CONTA
R LAS BUENAS Y SEGURAS CAPTURAS
QUE AMENAZAN LAS PIEZAS HUMANAS
 230 LET CSAFE=0: REM PARA CONTA
R LAS CAPTURAS SEGURAS QUE NO AM
ENAZAN LAS FICHAS
 240 LET CCAPTURE=0: REM PARA CO
NTAR OTRAS CAPTURAS ENCONTRADAS
 250 FOR J=1 TO 3
 260 LET G(J)=0: REM VACIO, ALMA
CENAR CAPTURA SEGURA
 270 LET S(J)=0
 280 LET T(J)=0: REM VACIO,OTRA
CAPTURA SEGURA
 290 NEXT J
 300 FOR J=80 TO 30 STEP -10
 310 FOR K=1 TO 8
 320 IF A(J+K)<>C THEN  GO TO 39
0: REM OMITIR LA EVALUACION SI N
O HAY NINGUNA PIEZA DE LA MAQUIN
A AHI
 330 REM ** CAPTURA A LA DERECHA
  **
 340 LET X=J+K-9: LET Y=J+K-18:
LET Z=J+K-27: LET M=-11
 350 IF A(X)=H AND A(7)=B THEN
GO SUB 700: REM CAPTURA ENCONTRA
DA
 360 REM ** CAPTURA A LA IZQUIER
DA **
 370 LET X=J+K-11: LET Y=J+K-22:
 LET Z=J+K-33: LET M=-9
 380 IF A(X)=H AND A(Y)=B THEN
GO SUB 700: REM CAPTURA ENCONTRA
DA
 390 NEXT K
 400 NEXT J
 410 IF GSAFE+CSAFE+CCAPTURE=0 T
HEN  GO TO 980: REM CAPTURA NO E
NCONTRADA
 420 REM ** AHORA ESCOGE LA CAPT
URA A HACER **
 430 PRINT : FOR I=1 TO 10: BEEP
 .3,I: BEEP .2,12: NEXT I: PRINT
 TAB 8;">> CAPTURA ENCONTRADA"
 440 PAUSE 600
 450 IF GSAFE<>0 THEN  GO TO 500
 460 IF CSAFE<>0 THEN  GO TO 670
 470 REM ** ELECCION DE CAPTURAS
 GENERALES **
 480 LET MOVE=T(INT (RND*CCAPTUR
E)+1)
 490 GO TO 540
 500 REM ELIGE
 510 REM SELECCION DE MOVIMIENTO
S ALMACENADOS
 520 LET MOVE=G(INT (RND*GSAFE)+
1)
 530 REM ** MOVER **
 540 LET START=INT (MOVE/100)
 550 LET ED=MOVE-100*START
 560 LET A(START)=B
 570 LET A(START-ED)=B
 580 LET A(START-2*ED)=C
 590 LET CS=CS+1
```

```
 600 REM * COMPROBAR PARA PUNTUA
CION ADICIONAL SI CAE FICHA EN F
ILA TRASERA *
 610 IF START-2*ED>18 THEN  RETU
RN
 620 LET A(START-2*ED)=B
 630 LET CS=CS+1
 640 PRINT "HE CAPTURADO Y DEJAD
O EN ";START-2*ED;" EN LA FILA D
E ATRAS"
 650 PAUSE 600
 660 RETURN
 670 REM ** CAPTURA SEGURA **
 680 LET        MOVE=S(INT (RND*C
SAFE)+1)
 690 GO TO 540
 700 REM ** COMPROBAR CAPTURA PR
OPUESTA PARA SEGURIDAD **
 710 REM ** COMPROBAR CUADRADO D
E ABAJO EN LA MISMA DIRECCION QU
E EL PROPOSITO DEL MOVIMIENTO **

 720 PRINT J+K;" A ";Y;" CAPTURA
NDO EN ";X
 730 PAUSE 400
 740 IF A(Z)=H THEN  GO TO 920:
REM ALMACENAR COMO UNA NUEVA CAP
TURA SEGURA
 750 REM ** COMPROBAR CUADRADO E
N LA OTRA DIRECCION DEL PROPOSIT
O DEL MOVIMIENTO **
 760 IF A(Y+M)=H AND A(Y-M)=B TH
EN  GO TO 920
 770 REM AHORA COMPRUEBA SI DEJA
RA UNA PIEZA EXPUESTA AL MOVER
 780 IF A(J+K+M)=H AND A(J+K+2*M
)=H THEN  GO TO 920
 790  REM ** SI SE ALCANZA ESTE
PUNTO, ENTONCES CAPTURA SEGURA *
 800 REM ** ALMACENA ESTE MOVIMI
ENTO **
 810 LET CSAFE=CSAFE+1
 820 LET S(CSAFE)=100*(J+K)+20+M
 830 REM ** MIRA A VER SI ES UN
BUEN MOVIMIENTO
 840 LET CHECK=GSAFE
 850 IF Y+2*M<1 THEN  RETURN
 860 IF A(Y+M)=H AND A(Y-(20+M))
<>B AND A(Y+2*M)=B THEN  LET GSA
FE=GSAFE+1
 870 IF CHECK=GSAFE THEN  RETURN
 : REM ** ESTE NO ES UN BUEN MOV
IMIENTO **
 880 REM * ALMACENAR BUEN MOVIMI
ENTO *
 890 PRINT "ESTOY CONSIDERANDO "
;J+K;" A ";M+20+J+K
 900 LET G(GSAFE)=100*(J+K)+20+M
 910 RETURN
 920 REM ALMACENAR CAPTURA NO SE
GURA
 930 LET CCAPTURE=CCAPTURE+1
 940 PRINT "ESTOY CONSIDERANDO "
;J+K;" A ";M+20+J+K
 950 LET T(CCAPTURE)=100*(J+K)+2
0+M
 960 RETURN
 970 REM ***********************
 980 REM * MOVER PARA PROTEGER P
IEZAS BAJO AMENAZA **
 990 LET MOVE=0
1000 LET J=80
1010 LET K=1
1020 LET Q=J+K
1030 IF A(Q)<>C THEN  GO TO 1110
: REM NO CONSIDERAR ESTE CUADRAD
O, NO MOVIMIENTO COMP
1040 IF A(Q+9)=B AND A(Q-9)=H AN
D A(Q+18)=C THEN  LET MOVE=100*(
Q+18)+Q+9
1050 REM EJECUCION ALEATORIA EN
CASO DE EXPOSICION DE OTRA
1060 IF MOVE<>0 AND A(Q-2)=H AND
 A(Q+20)=B AND RND>.5 THEN  GO T
O 1510
1070 IF A(Q+9)=B AND A(Q-9)=H AN
D A(Q+20)=C THEN  LET MOVE=100*(
Q+20)+Q+9: GO TO 1510
1080 IF A(Q+11)=B AND A(Q-11)=H
AND A(Q+22)=C THEN  LET MOVE=100
*(Q+22)+Q+11
1090 IF MOVE<>0 AND A(Q+2)=H AND
 A(Q+22)=B AND RND>.5 THEN  GO T
O 1510
1110 IF K<8 THEN  LET K=K+1: GO
TO 1020
1120 IF J>10 THEN  LET J=J-10: G
O TO 1010
1130 REM *********************
1140 REM ** NO ENCONTRADA CAPTUR
A. BUSCAR MOVIMIENTO PARA "DESAP
ARECER" EN LA FILA DE ATRAS ****
********************************
1150 LET MOVE=0
1160 REM MOVIMIMIENTO INDESEABLE
. SUSTITUIR POR OTRO MEJOR
1170 IF A(22)=C AND A(11)=B THEN
  LET MOVE=22
1180 IF A(28)=C AND A(17)=B THEN
  LET MOVE=28
1190 IF A(22)=C AND A(13)=B THEN
  LET MOVE=22
1200 IF A(26)=C AND A(17)=B THEN
  LET MOVE=26
1210 IF A(26)=C AND A(15)=B THEN
  LET MOVE=26
1220 IF A(24)=C AND A(15)=B THEN
  LET MOVE=24
```

```
1230 IF A(24)=C AND A(13)=B THEN
 LET MOVE=24
1240 IF MOVE=0 THEN  GO TO 1310
1250 PRINT : PRINT  INK 4; BRIGH
T 1;"ESTOY MOVIENDO SOBRE LA LIN
EA TRASERA DESDE ";MOVE
1260 PAUSE 500
1270 LET A(MOVE)=B
1280 LET CS=CS+1
1290 RETURN
1300 REM ***********************
1310 REM MOVIMIENTOS NO SEGUROS
1320 LET CMOVE=0: REM ** CUENTA
MOVIMIENTOS ENCONTRADOS **
1330 FOR J=80 TO 30 STEP -10
1340 FOR J=1 TO 8
1350 IF A(J+K)<>C THEN  GO TO 14
60
1360 LET X=J+K-9: LET Y=J+K-18:
LET Z=J+K-20
1370 LET Q=J+K+2
1380 IF A(X)<>B THEN  GO TO 1460
1390 IF A(Y)=H OR A(Z)=H AND A(Q
)=B THEN  GO TO 1460
1400 GO SUB 1560
1410 LET X=J+K-11: LET Y=K+J-22:
 LET Z=J+K-20
1420 LET Q=J+K-2
1430 IF A(X)<>B THEN  GO TO 1460
1440 IF A(Y)=H OR A(Z)=H AND A(Q
)=B THEN  GO TO 1460
1450 GO SUB 1560
1460 NEXT K
1470 NEXT J
1480 IF CMOVE=0 THEN  GO TO 1630
1490 REM ** MOVER **
1500 LET MOVE=T(INT (RND*CMOVE)+
1)
1510 LET START=INT (MOVE/100)
1520 LET ED=MOVE-100*START
1530 LET A(START)=B
1540 LET A(ED)=C
1550 RETURN
1560 REM ** ALMACENA MOVIMIENTO
**
1570 LET CMOVE=CMOVE+1
1580 PRINT "ESTOY CONSIDERANDO "
;J+K;" A ";X
1590 PAUSE 400
1600 LET T(CMOVE)=100*(J+K)+X
1610 RETURN
1620 REM *******************
1630 REM MOVIMIENTO ALEATORIO DE
 NO CAPTURA
1640 INK 6: PRINT "BUSCANDO MOVI
MIENTO ALEATORIO LEGAL"
1650 INK 7: LET L=0
1660 LET L=L+1
1670 LET J=10*INT (RND*8+1)
1680 LET K=INT (RND*8+1)
1690 IF A(J+K)=C THEN  GO TO 172
0
1700 IF L<200 THEN  GO TO 1660
1710 PRINT : FOR I=1 TO 4: BEEP
.5,I: BEEP .2,I+10: BEEP .3,I+5:
 NEXT I: PRINT "ABANDONO": STOP
1720 IF A(J+K-9)=B THEN  LET MOV
E=100*(J+K)+J+K-9: GO TO 1510
1730 IF A(J+K-11)=B THEN  LET MO
VE=100*(J+K)+J+K-11: GO TO 1510
1740 GO TO 1700
1750 REM ********************
1760 REM PINTAR TABLERO
1770 CLS
1780 PRINT
1790 PRINT  INK 6; BRIGHT 1;"MAQ
UINA  "; FLASH 1;CS; FLASH 0;"
 HUMANO "; FLASH 1;HS
1800 PRINT
1810 PRINT "      12345678 "
1820 PRINT "      --------"
1830 FOR J=80 TO 10 STEP -10
1840 PRINT "     ";J/10;
1850 FOR K=1 TO 8
1860 INK 4: PRINT  PAPER 5; INK
0;CHR$ (A(J+K));
1870 NEXT K
1875 INK 7
1880 PRINT J/10
1890 NEXT J
1900 PRINT "      --------"
1910 PRINT "      12345678 "
1920 PRINT
1930 RETURN
1940 REM ***********************
1950 REM ** ACEPTAR MOV HUMANO *
1955 REM ***********************
1960 INPUT "MUEVE DESDE ";START
1970 IF A(START)<H THEN  GO TO 1
960
1980 INPUT "HASTA ";ED
1990 PRINT ''; BRIGHT 1;"DESDE "
;START;",HASTA ";ED: IF A(ED)<>B
 OR ABS (START-ED)>11 AND A((STA
RT+ED)/2)<>C THEN  GO TO 1980
2000 LET A(START)=B
2010 LET A(ED)=H
2020 IF ABS (START-ED)>11 THEN
LET A((START+ED)/2)=B: LET HS=HS
+1: BEEP .3,7: BEEP .2,6: BEEP 1
,15: PRINT  INK 7; BRIGHT 1; PAP
ER 0; FLASH 1;" BIEN HECHO "
2030 IF ED>80 THEN  LET A(ED)=B:
 LET HS=HS+1: PRINT "UNO MAS PAR
A TI"
2040 PAUSE 100
2050 RETURN
2060 REM ********************
```

```
2070 REM *** INICIALIZACION ***
2080 REM **********************
2090 CLS
2100 RANDOMIZE
2110 DIM A(110): REM TABLERO Y E
SPACIOS BLANCOS ALREDEDOR Y DELA
NTE DE EL
2120 DIM G(3): REM BUENA CAPTURA
 SEGURA ALMACENADA
2130 DIM S(3): REM ALMACENA CAPT
URA SEGURA
2140 DIM T(18): REM OTRAS CAPTUR
AS ALMACENADAS. TAMBIEN USADO PA
RA MOVIMIENTOS SEGUROS SIN CAPTU
RA
2150 LET E=CODE " ": REM CUADRAD
O BLANCO
2160 LET B=CODE ".": REM CUADRAD
O NEGRO
2170 LET C=CODE "C": REM PIEZA D
E LA MAQUINA
2180 LET H=CODE "H": REM PIEZA H
UMANA
2190 LET HS=0: REM PUNTUACION HU
MANA
2200 LET CS=0: REM PUNTUACION DE
 LA MAQUINA
2210 RESTORE 2290: REM ESTABLECE
R TABLERO AL PRINCIPIO
2220 FOR J=10 TO 80 STEP 10
2230 FOR K=1 TO 8
2240 READ X: LET A(J+K)=X
2250 NEXT K
2260 NEXT J
2270 RETURN
2280 REM ********************
2290 DATA 72,32,72,32,72,32,72,3
2
2300 DATA 32,72,32,72,32,72,32,7
2
2310 DATA 46,32,46,32,46,32,46,3
2
2320 DATA 32,46,32,46,32,46,32,4
6
2330 DATA 46,32,46,32,46,32,46,3
2
2340 DATA 32,46,32,46,32,46,32,4
6
2350 DATA 67,32,67,32,67,32,67,3
2
2360 DATA 32,67,32,67,32,67,32,6
7
```

BLOQUELANDIA

```
  10 REM ** BLOQUELANDIA **
  15 FLASH 0: PAPER 1: BORDER 1:
 INK 7: BRIGHT 1
  20 GO SUB 2470: REM INICIALIZA
CION
  30 REM ** IMPRIMIR EL CUADRADO
 **
  40 INK 4: CLS : PRINT : PRINT
  50 FOR X=5 TO 1 STEP -1
  60 PRINT TAB 8;
  70 FOR I=1 TO 6
  80 PRINT  INK 0; PAPER 3;CHR$
(A(X,I));
  90 NEXT I
 100 PRINT
 110 NEXT X
 120 INK 7: PRINT : PRINT
 125 BEEP 1,15: FOR I=1 TO 5: BE
EP .1,I: NEXT I
 130 INPUT A$: LET A$=A$+"
          "
 140 PRINT A$: PRINT
 150 IF LEN A$=16 THEN  STOP : R
EM TERMINA LA EJECUCION EN VEZ D
E TENER QUE PRESIONAR LA TECLA "
ENTER"
 160 IF A$( TO 10)="DONDE ESTA"
THEN  GO SUB 240
 170 IF A$( TO 11)="DIME LO QUE"
 THEN  GO SUB 1050
 180 IF A$( TO 8)="REVUELVE" THE
N  GO SUB 1280
 190 IF A$( TO 6)="PON EL" THEN
 GO SUB 1500
 200 PRINT : PRINT : PRINT ">>PU
LSA ""ENTER"" PARA CONTINUAR<<":
 PAUSE 300
 210 GO TO 40
 220 STOP
 230 REM ***************
 240 REM "DONDE ESTA EL"
 250 LET P=0
 260 LET B$=A$(22)
 270 IF B$="R"  OR  B$="V"  OR B
$="N"  OR  B$="A"  THEN  GO TO 3
30
 280 IF RND>.7 THEN  GO TO 300
 290 PRINT "LO SIENTO, NO TENGO
INFORMACION DE ESO": GO TO 310
 300 PRINT "NO TENGO DATOS CON L
OS QUE PODER RESPONDERTE"
 310 RETURN
 320 REM ******************
 330 LET M=CODE B$
 340 PRINT TAB 8;">DEJAME VER AH
ORA<"
 350 LET X=5
 360 LET Y=1
 370 IF A(X,Y)=M THEN  GO TO 410
 380 IF Y<6 THEN  LET Y=Y+1: GO
TO 370
 390 IF X>1 THEN  LET X=X-1: GO
TO 360
 400 GO TO 280
 410 IF X>1 THEN  GO TO 910: REM
 ESTA ENCIMA DE OTRO
 420 IF Y>1 THEN  GO TO 530: REM
 NO ESTA A LA IZQUIERDA
 430 REM **********************
 440 REM *** A LA IZQUIERDA ***
 450 PRINT "ESTA A LA IZQUIERDA"
 460 IF A(1,2)=46 THEN  PRINT "N
O HAY NADA INMEDIATAMENTE A SU D
ERECHA": GO TO 790
 470 LET Q=A(1,2)
 480 PRINT
 490 PRINT "A SU LADO PUEDO VER
EL BLOQUE"
 500 GO TO 2400
 520 GO TO 790
 530 IF Y<6 THEN  GO TO 650
 540 REM ********************
 550 REM *** A LA DERECHA ***
 560 PRINT
 570 PRINT "ESTA EN EL LADO DERE
CHO"
 580 IF A(1,5)=46 THEN  PRINT "N
O HAY NADA INMEDIATAMENTE A SU I
ZQUIERDA": GO TO 790
 590 PRINT "A SU IZQUIERDA PUEDO
 VER EL BLOQUE";
 600 LET Q=A(1,5)
 610 GO SUB 2400
 630 GO TO 790
 640 REM ********************
 650 REM *** EN EL CENTRO ***
 660 PRINT
 670 PRINT "ESTA EN LA CASILLA "
;Y;" DE LA IZQUIERDA"
 680 IF A(X,Y-1)=46 THEN  PRINT
"NO HAY NADA INMEDIATAMENTE A SU
 IZQUIERDA": GO TO 730
 690 LET Q=A(X,Y-1)
 700 PRINT "EL BLOQUE ";
 710 GO SUB 2400
 720 PRINT " ESTA": PRINT "INMED
IATAMENTE A SU IZQUIERDA"
 730 IF A(X,Y-1)=46 THEN  PRINT
```

```
"NO HAY NADA INMEDIATAMENTE A SU
 IZQUIERDA": GO TO 790
 740 LET Q=A(X,Y+1)
 750 PRINT : PRINT "PUEDO VER EL
 BLOQUE ";
 760 GO SUB 2400
 770 PRINT " A SU DERECHA, TOCAN
DOLE"
 780 REM ********************
 790 REM ** ALGO POR ENCIMA? **
 800 PRINT
 810 LET P=X
 820 IF X=5 THEN  GO TO 910
 830 IF A(X+1,Y)=46 THEN  PRINT
"NO HAY NADA ENCIMA DE EL": GO T
O 310
 840 PRINT : PRINT "ENCIMA ESTA
EL BLOQUE ";
 850 LET Q=A(X+1,Y)
 860 GO SUB 2400
 880 LET X=X+1
 890 GO TO 820
 900 REM ********************
 910 REM ** ENCIMA DE OTRO? **
 920 IF P<>0 THEN  LET X=P
 930 PRINT
 940 IF X=1 THEN  GO TO 310
 950 PRINT "ESTA INMEDIATAMENTE
 ";
 960 PRINT "ENCIMA DEL BLOQUE ";
 970 LET Q=A(X-1,Y)
 980 GO SUB 2400
1000 LET X=X-1
1010 IF X<2 THEN  GO TO 310
1020 GO TO 960
1030 RETURN
1040 REM ********************
1050 REM ** DIME LO QUE VES **
1060 PRINT "EMPEZANDO DESDE LA D
ERECHA"
1070 LET Y=6
1080 LET X=5
1090 IF A(X,Y)<>46 THEN  GO TO 1
150
1100 IF Y=1 AND X=1 THEN  PRINT
"FINALMENTE ";
1110 IF X=1 AND A(X,Y)=46 THEN
PRINT "...UN ESPACIO ";: IF Y>1
THEN  PRINT "LUEGO"
1120 IF X>1 THEN  LET X=X-1: GO
TO 1090
1130 IF Y>1 THEN  LET Y=Y-1: GO
TO 1080
1140 RETURN
1150 LET L=INT (RND*2)
1160 IF L=0 THEN  PRINT "ESTA EL
 BLOQUE ";: GO TO 1190
1170 IF L=1 THEN  PRINT "MIS SEN
SORES REGISTRAN EL BLOQUE ";: GO
 TO 1190
1180 PRINT "VEO EL BLOQUE ";
1190 LET Q=A(X,Y)
1200 GO SUB 2400
1220 IF X=1 THEN  GO TO 1130
1230 LET X=X-1
1240 PRINT "...Y MAS ABAJO DE EL
..."
1250 GO TO 1180
1260 RETURN
1270 REM ******************
1280 REM * REVUELVE BLOQUES *
1290 PRINT
1300 IF RND>.5 THEN  INK 3: PRIN
T TAB 7;"YA ERA HORA": GO TO 132
0
1310 INK 5: PRINT "ESTA BIEN QUE
 LE DEN A UNO LA OPORTUNIDAD ":
PRINT TAB 4;"DE HACER LO QUE QUI
ERE!!!"
1320 INK 7: FOR X=1 TO 5
1330 FOR Y=1 TO 6
1340 LET A(X,Y)=46
1350 NEXT Y
1360 NEXT X
1370 LET Y1=INT (RND*6)+1
1380 LET Y2=INT (RND*6)+1
1390 IF Y2=Y1 THEN  GO TO 1380
1400 LET Y3=INT (RND*6)+1
1410 IF Y3=Y1 OR Y2=Y3 THEN  GO
TO 1380
1420 LET Y4=INT (RND*6)+1
1430 IF Y3=Y4  OR Y4=Y1 OR Y2=Y4
 THEN  GO TO 1380
1440 LET A(1,Y1)=82
1450 LET A(1,Y2)=86
1460 LET A(1,Y3)=65
1470 LET A(1,Y4)=78
1480 RETURN
1500 REM "PON EL BLOQUE...ENCIMA
 DEL BLOQUE..."
1510 IF RND>.5 THEN  PRINT TAB 5
;"COMPRENDO": GO TO 1530
1520 PRINT TAB 8;"> DE ACUERDO"
1530 LET B$=A$(15): REM BLOQUE O
BJETO
1540 IF B$="R" THEN  LET L=38
1550 IF B$="A" THEN  LET L=38
1560 IF B$="V" THEN  LET L=39
1570 IF B$="N" THEN  LET L=41
1580 LET C$=A$(L)
1590 LET B=CODE B$
1600 LET C=CODE C$
1610 LET FLAG=C
1620 REM **ENCUENTRA EL BLOQUE B
$ **
1630 LET X=5
```

```
1640 LET Y=1
1650 IF A(X,Y)=B THEN  GO TO 174
0
1660 IF Y<6 THEN  LET Y=Y+1: GO
TO 1650
1670 IF X>1 THEN  LET X=X-1: GO
TO 1640
1680 PRINT "NO PUEDO ENCONTRAR E
L BLOQUE ";
1690 LET Q=B
1700 GO SUB 2400
1720 FOR T=1 TO 200: NEXT T
1730 RETURN
1740 LET R=X: LET S=Y
1750 REM ** EL BLOQUE ESTA EN R,
S **
1760 REM ** ESTA VACIA LA CASILL
A DE DESTINO? **
1770 IF A(R+1,S)=46 THEN  GO TO
1920: REM "SI"
1780 IF A(R+2,S)=46 THEN  LET TA
SK=1: GO TO 1800
1790 LET TASK=3: IF A(R+3,S)=46
THEN  LET TASK=2
1800 FOR W=TASK TO 1 STEP -1
1810 PRINT "DEBO MOVER EL BLOQUE
 ";
1820 LET Q=A(R+W,S)
1830 GO SUB 2400
1850 LET DE=INT (RND*6)+1
1860 IF DE=S OR A(1,DE)=C OR A(2
,DE)=C OR A(3,DE)=C THEN  GO TO
1850
1870 PRINT "ESTOY MOVIENDOLO A L
A COLUMNA ";DE
1880 LET L=1
1890 IF A(L,DE)=46 THEN  LET A(L
,DE)=A(R+W,S): LET A(R+W,S)=46:
GO TO 1910
1900 LET L=L+1: GO TO 1890
1910 NEXT W
1920 REM LA CASILLA DE DESTINO E
N R,S ESTA AHORA LIBRE
1930 REM ESTA LIBRE LA CASILLA D
E PARTIDA
1940 REM ENCUENTRA EL BLOQUE
1950 LET X=5
1960 LET Y=1
1970 IF A(X,Y)=C THEN  GO TO 207
0
1980 IF Y<6 THEN  LET Y=Y+1: GO
TO 1970
1990 IF X>1 THEN  LET X=X-1: GO
TO 1960
2000 PRINT "NO PUEDO ENCONTRAR E
L BLOQUE ";
2010 LET Q=C
2020 GO SUB 2400
2040 FOR J=1 TO 200: NEXT J
2050 RETURN
2060 REM ** SE HA ENCONTRADO C *
*
2070 LET T=X: LET U=Y: REM LOCAL
IZACION DE C
2080 IF A(T+1,U)=46 THEN  GO TO
2260
2090 IF A(T+2,U)=46 THEN  LET TA
SK=1: GO TO 2110
2100 IF A(T+3,U)=46 THEN  LET TA
SK=2
2110 LET DE=INT (RND*6)+1
2120 IF DE=U OR DE=S THEN  GO TO
 2110
2130 FOR W=TASK TO 1 STEP -1
2140 PRINT "AHORA MOVERE EL BLOQ
UE ";
2150 LET Q=A(T+W,U)
2160 GO SUB 2400
2180 PRINT
2190 PRINT "ESTOY MOVIENDO A LA
COLUMNA ";DE
2200 LET L=1
2210 IF A(L,DE)=46 THEN  LET A(L
,DE)=A(T+W,U): LET A(T+W,U)=46:
GO TO 2230
2220 LET L=L+1: GO TO 2210
2230 NEXT W
2240 REM LA CASILLA DE PARTIDA A
HORA ESTA LIBRE
2250 REM *** REALIZA EL MOVIMIEN
TO ***
2260 PRINT "AHORA ESTOY PONIENDO
 EL BLOQUE ";
2270 LET Q=A(R,S): LET Z=A(R,S)
2280 GO SUB 2400
2300 PRINT " ENCIMA DEL BLOQUE "
;
2310 IF A(T,U)=46 THEN  LET A(T,
U)=FLAG
2320 LET Q=A(T,U)
2330 GO SUB 2400
2350 LET A(R,S)=46
2360 LET A(T+1,U)=Z
2370 FOR J=1 TO 200: NEXT J
2380 RETURN
2390 REM *********************
2400 REM ** NOMBRE DEL COLOR **
2410 IF Q=CODE "R" THEN  PRINT "
ROJO ";
2420 IF Q=CODE "N" THEN  PRINT "
NARANJA ";
2430 IF Q=CODE "A" THEN  PRINT "
AZUL ";
2440 IF Q=CODE "V" THEN  PRINT "
VERDE ";
2450 RETURN
```

```
2460 REM *******************
2470 REM ** INICIALIZACION **
2480 CLS
2490 RANDOMIZE
2500 DIM A(5,6)
2510 FOR X=1 TO 5
2520 FOR Y=1 TO 6
2530 LET A(X,Y)=46
2540 NEXT Y
2550 NEXT X
2560 LET A(1,2)=CODE "R": REM BL
OQUE ROJO
2570 LET A(1,3)=CODE "N": REM BL
OQUE NARANJA
2580 LET A(1,4)=CODE "A": REM BL
OQUE AZUL
2590 LET A(1,5)=CODE "V": REM BL
OQUE VERDE
2600 RETURN
```

DOCTOR

```
  10 REM ** DOCTOR **
  15 BRIGHT 1: FLASH 0: PAPER 1:
 INK 7: BRIGHT 0: BORDER 1: INVE
RSE 0: CLS
  20 GO SUB 1140: REM INICIALIZA
CION
  30 PRINT : PRINT  INK 6; BRIGH
T 1;"---------------------------
-----"
  40 POKE 23658,200: PRINT ">";:
 INPUT X$: PRINT X$'
  50 IF X$="" THEN  STOP : REM T
ERMINAR APRETANDO "ENTER"
  60 PRINT
  70 IF X$=Z$ THEN  PRINT "POR F
AVOR, NO SE REPITA": GO TO 30
  80 LET Z$=X$: LET X$=X$+"
                               "
  90 IF X$( TO 5)="ADIOS" THEN
PRINT "ESPERO VERLE DE NUEVO": S
TOP
 100 REM ** BUSQUEDA DE FRASES C
LAVE AL PRINCIPIO DE LA INTRODUC
CION **
 110  LET L=0
 120  LET L=L+1
 130  LET LN=CODE C$(L,1)-40
 140 IF X$( TO LN-1)=C$(L,2 TO L
N) THEN  GO TO 360: REM SE HA EN
CONTRADO LA FRASE CLAVE
 150 IF L<K THEN  GO TO 120
 160 REM ** EL PROGRAMA LLEGA AQ
UI EN EL CASO DE QUE NO SE HAYA
ENCONTRADO UNA FRASE CLAVE AL PR
INCIPIO DE X$ **
 170 REM ** AHORA SE BUSCAN PALA
BRAS CLAVE DENTRO DE TODA LA FRA
SE INTRODUCIDA **
 180 LET X$=" "+X$+" "
 190 LET M=LEN X$
 200 LET L=0
 210 LET L=L+1
 220 IF L=M-1 THEN  GO TO 800: R
EM NO SE HA ENCONTRADO PALABRA C
LAVE
 230 IF X$(L)=" " THEN  GO TO 25
0
 240 GO TO 210
 250 LET X=L+1
 260 LET Y=0
 270 LET Y=Y+1
 280 IF X$(X+Y)=" " THEN  LET Q$
=X$(X TO X+Y): GO TO 300
 290 GO TO 270
 300 LET N=0
 310 LET N=N+1
 320 IF Q$( TO LEN Q$)=C$(N,2 TO
 CODE C$(N,1)-40) THEN  GO TO 71
0: REM SE HA ENCONTRADO UNA PALA
BRA CLAVE
 330 IF N<K THEN  GO TO 310
 340 GO TO 210
 350 REM ***********************
 360 REM ** SE HA ENCONTRADO UNA
 FRASE CLAVE AL PRINCIPIO DE LA
INTRODUCCION **
 370 LET T=INT (RND*3)+1
 380 IF T=1 THEN  LET G$=D$(L)
 390 IF T=2 THEN  LET G$=E$(L)
 400 IF T=3 THEN  LET G$=F$(L)
 410 REM ** SE VERIFICA SI LA FR
ASE ACABA EN UN ASTERISCO. DE SE
R ASI HAY QUE ANADIRLA ALGO MAS
**
 420 LET FLAG=0
 425 FOR R=65 TO 1 STEP -1: IF G
$(R)=" " THEN  NEXT R
 430 IF G$(R)="*" THEN  LET FLAG
=1: LET G$=G$( TO R-1)
 440 PRINT G$;" ";
 450 IF FLAG=0 THEN  GO TO 30: R
EM NO HAY QUE ANADIR NADA
 460 REM ** AHORA SE UTILIZA LA
FRASE INTRODUCIDA **
 465 FOR R=LEN X$ TO 1 STEP -1:
IF X$(R)=" " THEN  NEXT R
 466 LET X$=X$( TO R)
 470 LET X$=" "+X$(LN+1 TO LEN X
$)+" "
 480 REM ***********************
 490 REM ** CAMBIOS DE CONJUGACI
ON **
 500 REM * TAMBIEN SE BUSCA "MI"
 PARA LA CLAVE "MIFLAG"(K$) **
 510 LET LN=LEN X$
 520 LET M=0
 530 LET M=M+1
 540 IF M=LN THEN  GO TO 30
 550 IF X$(M)=" " THEN  GO TO 57
0
 560 GO TO 530
 570 LET X=M+1
 580 LET Y=0
 590 LET Y=Y+1
 595 IF LEN X$<X+Y THEN  LET X$=
X$+" "
 600 IF X$(X+Y)=" " THEN  LET Q$
=X$(X TO X+Y-1): GO TO 630
```

```
 610 IF X+Y>250 THEN  GO TO 530
 620 GO TO 590
 630 LET MN=0
 640 LET MN=MN+1
 650 IF Q$="MI" AND K$="" THEN
LET K$=X$(X+3 TO LEN X$-4)
 660 IF Q$=A$(MN,2 TO CODE A$(MN
,1)-40) THEN  PRINT B$(MN,2 TO C
ODE B$(MN,1)-40);" ";: GO TO 530
 670 IF MN<KK THEN  GO TO 640
 680 PRINT Q$;" ";
 690 GO TO 530
 700 REM ***********************
 710 REM ENCUENTRA PALABRA CLAVE
 720 LET T=INT (RND*3)+1
 730 LET Q$=""
 740 IF T=1 THEN  LET Q$=D$(N)
 750 IF T=2 THEN  LET Q$=E$(N)
 760 IF T=3 THEN  LET Q$=F$(N)
 770 IF Q$(LEN Q$)="*" THEN  PRI
NT Q$: GO TO 30
 780 REM ** SE PASA A LA SIGUIEN
TE SECCION SI LA PALABRA CLAVE S
E CONSIDERA NO VALIDA **
 790 REM ***********************
 800 REM RESPUESTAS ALEATORIAS/N
O CLAVES
 820 LET T=INT (RND*8)+1
 850 IF T=1 THEN  PRINT "QUE LE
SUGIERE A USTED?": GO TO 30
 870 IF T=2 THEN  PRINT "NO ESTO
Y SEGURO DE COMPRENDER ESO POR C
OMPLETO": GO TO 30
 890 IF T=3 THEN  PRINT "NO SEA
TAN ESCUETO, POR FAVOR   ACLAREL
O UN POCO MAS": GO TO 30
 910 IF T=4 THEN  PRINT "ESO ES
MUY INTERESANTE": GO TO 30
 930 IF T=5 THEN  PRINT "BIEN,BI
EN...POR FAVOR CONTINUE...": GO
TO 30
 950 IF T=6 THEN  PRINT "POR QUE
?": GO TO 30
 970 IF T=7 THEN  PRINT "Y ENTON
CES?": GO TO 30
 990 IF T=8 THEN  PRINT "YA VEO.
..POR FAVOR SIGA HABLANDO MAS SO
BRE ELLO.": GO TO 30
1010 REM ** UTILIZACION DE "MIFL
AG" **
1020 LET T=INT (RND*8)+1
1030 IF T=1 THEN  PRINT "HABLEMO
S MAS SOBRE SU ";K$
1040 IF T=2 THEN  PRINT "AL PRIN
CIPIO ME HABLO DE SU ";K$;".CUEN
TEME ALGO MAS."
1050 IF T=3 THEN  PRINT "TIENE A
LGO QUE VER ";K$;"?"
1060 IF T=4 THEN  PRINT "EXISTE
ALGUNA CONEXION CON ";K$;"?"
1070 IF T=5 THEN  PRINT "POR QUE
 NO RETROCEDEMOS UN POCO Y HABLA
MOS MAS DE SU ";K$;"?"
1080  IF T=6 THEN  PRINT "LE SUG
IERE QUE EXISTA ALGUNA CONEXION
ENTRE ESO Y SU ";K$;"?"
1090 IF T=7 THEN  PRINT "PREFERI
RIA HABLAR SOBRE SU ";K$;"?"
1100 IF T=8 THEN  PRINT "CREO QU
E EL PREOCUPARSE SOBRE SU ";K$;"
 NO LE CONDUCE A NADA"
1110 IF RND>.7 THEN  LET K$=""
1120 GO TO 30
1130 REM *******************
1140 REM ** INICIALIZACION **
1150 RANDOMIZE
1160 CLS
1180 DIM A$(16,20): DIM B$(16,20
): REM CONJUGACIONES
1190 DIM C$(45,16): DIM D$(45,65
): DIM E$(45,65): DIM F$(45,65):
 REM PALABRAS CLAVE Y RESPUESTAS
1200 LET Z$="": REM PARA FINALIZ
AR LAS REPETICIONES
1210 LET K$="": REM "MIFLAG"
1220 PRINT : PRINT
1230 PRINT  FLASH 1; PAPER 1; IN
K 5;"EL DOCTOR ESTA EN LA CONSUL
TA...POR FAVOR ESPERE"
1240 LET KK=0
1250 LET KK=KK+1
1260 READ A$(KK),B$(KK)
1261 FOR R=14 TO 1 STEP -1: IF A
$(KK,R)=" " THEN  NEXT R
1262 LET A$(KK)=CHR$ (R+41)+A$(K
K, TO 19)
1263 FOR R=14 TO 1 STEP -1: IF B
$(KK,R)=" " THEN  NEXT R
1264 LET B$(KK)=CHR$ (R+41)+B$(K
K, TO 19)
1270 IF B$(KK,2)="*" THEN  GO TO
 1290
1280 GO TO 1250
1290 LET K=0
1300 LET K=K+1
1310 READ C$(K),D$(K),E$(K),F$(K
)
1311 FOR R=10 TO 1 STEP -1: IF C
$(K,R)=" " THEN  NEXT R
1313 LET C$(K)=CHR$ (R+41)+C$(K,
 TO 14)
1320 IF F$(K,1)="*" THEN  GO TO
1340
1330 GO TO 1300
1340 CLS
```

```
1350 PRINT "BIENVENIDO A OTRA SE
SION"
1360 PRINT "CON EL DOCTOR..."
1380 PRINT '"HOLA..."
1390 RETURN
1400 REM ***********
1410 REM ** DATOS **
1420 REM ** CONJUGACIONES **
1430 DATA "SOY","ES","ERA","ERA"
,"YO","USTED","MI","SU","MIS","S
US"
1440 DATA "USTED TIENE","YO TENG
O","SU","MI","ES","SOY","ME","LE
"
1450 DATA "USTED ESTA","YO ESTOY
","SUYO","MIO","USTED","MI","MI"
,"USTED","*","*"
1460 REM ** PALABRAS CLAVE/FRASE
S DE RESPUESTA **
1470 DATA "NECESITO"
1480 DATA "POR QUE NECESITA*"
1490 DATA "NO SERIA REALMENTE PR
OVECHOSO SI USTED*"
1500 DATA "ESTA SEGURO DE QUE NE
CESITA*"
1510 DATA "POR QUE NO*"
1520 DATA "PIENSA REALMENTE QUE
YO NO*"
1530 DATA "QUIZAS FINALMENTE YO*
"
1540 DATA "REALMENTE QUIERE QUE
YO*"
1550 DATA "POR QUE NO PUEDO"
1560 DATA "CREE QUE USTED SERIA
CAPAZ DE*"
1570 DATA "POR QUE NO PUEDE*"
1580 DATA "QUIZAS NO LO HAYA INT
ENTADO LO SUFICIENTE"
1590 DATA "ESTA USTED"
1600 DATA "POR QUE ESTA INTERESA
DO EN SABER SI YO SOY O NO*"
1610 DATA "LO PREFERIRIA SI NO F
UERA*"
1620 DATA "QUIZAS ALGUNAS VECES
SUENE QUE SOY*"
1630 DATA "NO PUEDO"
1640 DATA "COMO SABE QUE NO PUED
E*"
1650 DATA "LO HA INTENTADO?"
1660 DATA "QUIZAS ,AHORA, USTED
PUEDA"
1670 DATA "YO ESTOY"
1680 DATA "ACUDIO A MI PORQUE ES
TA*"
1690 DATA "CUANTO TIEMPO HA ESTA
DO*"
1700 DATA "CREE QUE ES NORMAL ES
TAR*"
1710 DATA "SOY"
1720 DATA "DISFRUTA SIENDO*"
1730 DATA "POR QUE ME DICE QUE E
S*"
1740 DATA "POR QUE ES USTED*"
1750 DATA "QUIERO"
1760 DATA "QUE SIGNIFICARIA PARA
 USTED SI*"
1770 DATA "POR QUE QUIERE*"
1780 DATA "QUE SUPONDRIA EN SU V
IDA SI USTED CONSIGUIERA*"
1790 DATA "QUE"
1800 DATA "POR QUE PREGUNTA ESO?
"
1810 DATA "QUE RESPONDERIA PARA
AYUDARSE?"
1820 DATA "QUE PIENSA USTED?"
1830 DATA "COMO"
1840 DATA "COMO ARREGLARIA ESO?"
1850 DATA "LO MEJOR SERIA QUE RE
SPONDIESE A ESO USTED MISMO"
1860 DATA "QUE ES LO QUE REALMEN
TE ESTA PREGUNTANDO?"
1870 DATA "QUIEN"
1880 DATA "SE HACE A MENUDO ESAS
 PREGUNTAS?"
1890 DATA "QUE RESPUESTA LE DEJA
RIA TRANQUILO?"
1900 DATA "QUIEN CREE USTED QUE*
"
1910 DATA "DONDE"
1920 DATA "ESA ES UNA PREGUNTA B
ASTANTE INGENUA"
1930 DATA "NECESITA REALMENTE SA
BER DONDE?"
1940 DATA "QUE SUPONDRIA PARA US
TED SI YO LE DIJERA DONDE*"
1950 DATA "CUANDO"
1960 DATA "COMO PODRIA SABER CUA
NDO*"
1970 DATA "NO ESTAMOS AQUI PARA
DISCUTIR SOBRE EL TIEMPO"
1980 DATA "LAS COSAS SUELEN OCUR
RIR CASI SIEMPRE EN EL MOMENTO O
PORTUNO"
1990 DATA "POR QUE"
2000 DATA "POR QUE NO ME DICE LA
 VERDADERA RAZON POR LA QUE*"
2010 DATA "QUE ES LO QUE HA DICH
O QUE ME AYUDARIA A DECIRLE POR
QUE*"
2020 DATA "NECESITA REALMENTE SA
BER POR QUE*"
2030 DATA "PORQUE"
2040 DATA "ES ESA LA VERDADERA R
AZON?"
2050 DATA "QUE OTRAS RAZONES REC
UERDA?"
```

```
2060 DATA "QUE MAS PUEDE EXPLICA
R ESO?"
2070 DATA "LO SIENTO"
2080 DATA "EN QUE OTRAS CIRCUNST
ANCIAS SE DISCULPA?"
2090 DATA "HAY MUCHAS OCASIONES
EN LAS QUE NO ES NECESARIO DISCU
LPARSE"
2100 DATA "QUE SENSACION EXPERIM
ENTA CUANDO TIENE QUE DISCULPARS
E?"
2110 DATA "HOLA"
2120 DATA "HOLA...ME ALEGRO DE V
ERLE"
2130 DATA "HOLA...ME ALEGRO QUE
SE HAYA DEJADO CAER HOY POR AQUI
"
2140 DATA "COMO ESTA USTED?...TE
NIA GANAS DE QUE CHARLARAMOS DE
NUEVO"
2150 DATA "OIGA!"
2160 DATA "ME ALEGRO DE VERLE HO
Y POR AQUI"
2170 DATA "ME ALEGRO DE QUE HAYA
 VENIDO. TENEMOS UN MOTON DE COS
AS DE LAS QUE HABLAR"
2180 DATA "ATIENDAME AHORA.RELAJ
ESE...Y HABLEMOS UN POCO SOBRE S
U SITUACION"
2190 DATA "PUEDE SER"
2200 DATA "PARECE VACILAR UN POC
O"
2210 DATA "PARECE BASTANTE INDEC
ISO"
2220 DATA "EN QUE OTRAS CIRCUNST
ANCIAS SE MUESTRA TAN INDECISO?"
2230 DATA "NO"
2240 DATA "POR QUE SE MUESTRA TA
N NEGATIVO SOBRE ESO?"
2250 DATA "ESTA USTED DICIENDO Q
UE ESO NO ES SER NEGATIVO?"
2260 DATA "ES USTED BASTANTE IMP
ULSIVO.QUE LE SUGIERE ESO?"
2270 DATA "SIEMPRE"
2280 DATA "POR FAVOR, DEME UN EJ
EMPLO CONCRETO"
2290 DATA "NO LE PARECE QUE 'SIE
MPRE' ES UN POCO EXAGERADO?"
2300 DATA "CUANDO?"
2310 DATA "CREO"
2320 DATA "DUDA*"
2330 DATA "PIENSA REALMENTE ESO?
"
2340 DATA "PERO NO ESTA SEGURO*"
2350 DATA "AMIGO"
2360 DATA "POR QUE TOCA AHORA EL
 TEMA DE LOS AMIGOS?"
2370 DATA "CUAL ES SU MEJOR RECU
ERDO DE UN AMIGO?"
2380 DATA "HABLEME MAS DE SUS AM
ISTADES..."
2390 DATA "AMIGOS"
2400 DATA "EN QUE MODO LE AFECTA
N LAS REACCIONES DE SUS AMIGOS?"
2410 DATA "QUE LE HIZO COMENZAR
A HABLAR DE LOS AMIGOS,PRECISAME
NTE AHORA?"
2420 DATA "DE QUE MODO LE ENGANA
N SUS AMIGOS?"
2430 DATA "SI"
2440 DATA "QUE TIPO DE MENTIRAS
LE MOLESTAN?"
2450 DATA "ES USTED BASTANTE POS
ITIVO"
2460 DATA "PARECE MUY SEGURO DE
ELLO"
2470 DATA "ORDENADOR"
2480 DATA "QUE ES LO QUE SIENTE
SENTADO AHI Y HABLANDOME COMO A
UNA PERSONA?"
2490 DATA "LO PIENSA SOBRE MI EN
 PARTICULAR?"
2500 DATA "QUE ASPECTOS DE LOS O
RDENADORES LE INTERESAN MAS?"
2510 DATA "ESTA"
2520 DATA "PIENSA QUE ESTAR *"
2530 DATA "EN QUE CIRCUNSTANCIAS
 ESTARIA*"
2540 DATA "BIEN PODRIA SER *"
2550 DATA "ES"
2560 DATA "QUE GRADO DE CERTEZA
DEPOSITARIA EN ELLO SIENDO*"
2570 DATA "ESTA USTED SEGURO DE
QUE ES*"
2580 DATA "QUE SENSACIONES EXPER
IMENTARIA SI LE DIJERA QUE,PROBA
BLEMENTE NO ES *"
2590 DATA "PUEDE"
2600 DATA "QUE LE HACE PENSAR QU
E NO PUEDO*"
2610 DATA "NO PIENSA QUE PUEDO*"
2620 DATA "QUIZAS LE GUSTARIA SE
R CAPAZ DE*"
2630 DATA "PUEDO"
2640 DATA "TAL VEZ NO QUIERA*"
2650 DATA "LE GUSTARIA SER CAPAZ
 DE*"
2660 DATA "LO DUDO"
2670 DATA "SOY"
2680 DATA "POR QUE PIENSA QUE SO
Y*"
2690 DATA "LE COMPLACE CREER QUE
 SOY*"
2700 DATA "TAL VEZ LE GUSTARIA S
ER*"
```

```
2710 DATA "ESTA"
2720 DATA "POR QUE PIENSA QUE ES
TOY*"
2730 DATA "LE COMPLACE CREER QUE
 ESTOY*"
2740 DATA "POR QUE DICE QUE ESTO
Y*"
2750 DATA "YO NO"
2760 DATA "REALMENTE NO*"
2770 DATA "POR QUE NO*"
2780 DATA "LE GUSTARIA SER CAPAZ
 DE*"
2790 DATA "SIENTO"
2800 DATA "HABLEME MAS SOBRE ESO
S SENTIMIENTOS"
2810 DATA "SIENTE A MENUDO"
2820 DATA "DISFRUTA SINTIENDOSE*
"
2830 DATA "SIENTO"
2840 DATA "VAMOS A EXAMINAR ESA
AFIRMACION MAS DETENIDAMENTE"
2850 DATA "SE SIENTE NORMALMENTE
 DE ESE MODO?"
2860 DATA "QUE EMOCIONES LE DESP
IERTAN ESOS SENTIMIENTOS?"
2870 DATA "TENGO"
2880 DATA "POR QUE ME DICE QUE T
IENE*"
2890 DATA "PARA MI ESTA CLARO QU
E TIENE*"
2900 DATA "COMO PODRIA AYUDARLE
CON*"
2910 DATA "PODRIA"
2920 DATA "PODRIA EXPLICARME POR
 QUE SERIA*"
2930 DATA "A QUIEN MAS LE HA DIC
HO QUE USTED PODRIA*"
2940 DATA "QUE SEGURIDAD TIENE D
E QUE PODRIA"
2950 DATA "HAY"
2960 DATA "POR SUPUESTO QUE HAY*
"
2970 DATA "ES COMO SI HUBIERA*"
2980 DATA "LE GUSTARIA QUE HUBIE
RA*"
2990 DATA "MI"
3000 DATA "SU"
3010 DATA "YA VEO, SU*"
3020 DATA "QUE SIGNIFICA PARA US
TED EL QUE SU*"
3030 DATA "USTED"
3040 DATA "ESTA SESION ES PARA A
YUDARLE....NO PARA QUE ANDEMOS D
ISCUTIENDO"
3050 DATA "QUE LE IMPULSO A DECI
R ESO SOBRE MI?"
3060  DATA "RECUERDE QUE ANOTO T
ODO CUANTO DICE, PARA LUEGO PODE
R AYUDARLE"
3070 DATA "*","*","*","*"
```

TRADUCTOR

```
  10 REM *** TRADUCTOR ***
  15 POKE 23658,200
  20 GO SUB 9000: REM INICIALIZA
CION
  25 FLASH 0: BRIGHT 1: PAPER 1:
 INK 7: BRIGHT 0: BORDER 1: CLS
  30 INPUT "ESCRIBE LO QUE QUIER
AS TRADUCIR";A$: REM ACEPTA LA F
RASE DEL USUARIO
  35 BEEP .2,1
  40 IF A$="" THEN  STOP
  45 PRINT "?";A$
  50 INK 5: GO SUB 1000: INK 7:
REM TRADUCCION
  60 GO TO 30
 240 LET M=1
 980 STOP
 999 REM ****************
1000 REM ** TRADUCCION **
1002 PRINT TAB 2;"-->";
1005 LET L=LEN A$: LET K=0
1010 LET K=K+1
1015 IF K=L THEN  PRINT : PRINT
: RETURN
1020 LET B$=" "+A$+" ": IF B$(K)
=" " THEN  GO TO 1200
1030 GO TO 1010
1200 LET X=K+1
1210 LET Y=0
1220 LET Y=Y+1
1230 IF B$(X+Y)=" " THEN  LET Q$
=B$(X TO X+Y): GO TO 1240
1235 GO TO 1220
1240 LET M=0
1250 LET M=M+1
1260 IF Q$<>E$(M, TO LEN Q$) THE
N  GO TO 1265
1261 FOR G=15 TO 1 STEP -1: IF F
$(M,G)=" " THEN  NEXT G
1262 PRINT F$(M,1 TO G);" ";: GO
 TO 1280
1265 IF M<COUNT THEN  GO TO 1250
1270 LET Q$=Q$( TO LEN Q$-1): PR
INT  FLASH 1;Q$;: PRINT " ";
1280 GO TO 1010
8990 REM ********************
9000 REM ** INICIALIZACION **
9005 CLS
9010 DIM E$(110,20): REM ALMACEN
A LAS PALABRAS EN ESPANOL
9020 DIM F$(110,20): REM ALMACEN
A LAS PALABRAS EN INGLES
9100 LET COUNT=0
9105 LET COUNT=COUNT+1
9110 READ E$(COUNT),F$(COUNT)
9120 IF F$(COUNT,1)<>"*" THEN  G
O TO 9105
9490 RETURN
9500 REM ** DATOS **
9510 DATA "YO","I","TU","YOU","U
STED","YOU","ELLA","SHE","ELLO",
"IT","NOSOTROS","WE","VOSOTROS",
"YOU","ELLOS","THEY"
9520 DATA "EL","THE","LA","THE",
"LOS","THE","LAS","THE","UN","A"
,"UNA","A"
9530 DATA "MIO","MINE","TUYO","Y
OURS","NUESTRO","OUR","VUESTRO",
"YOUR"
9540 DATA "SOY","AM","ERES","ARE
","ES","IS","SOMOS","ARE","SOIS"
,"ARE","SON","ARE"
9550 DATA "TENGO","HAVE","TIENES
","HAVE","TIENEN","HAVE"
9560 DATA "HE","HAVE","HAS","HAV
E","HA","HAS","HEMOS","HAVE","HA
BEIS","HAVE","HAN","HAVE"
9570 DATA "COMIDO","EATEN","BEBI
DO","DRUNK","TENIDO","HAD","SIDO
","BEEN"
9580 DATA "Y","AND","E","AND","O
","OR","U","OR","SI","YES","NO",
"NO","DE","OF","CON","WITH","EN"
,"IN"
9590 DATA "BRAZO","ARM","PIERNA"
,"LEG","OJO","EYE","CABEZA","HEA
D","DERECHA","RIGHT","IZQUIERDA"
,"LEFT","IZQUIERDO","LEFT"
9600 DATA "GINEBRA","GIN","VINO"
,"WINE","GUISKY","WHISKY","CERVE
ZA","BEER","AGUA","WATER"
9610 DATA "PESCADO","FISH","POLL
O","CHICKEN","PATO","DUCK","MOST
AZA","MUSTARD","SAL","SALT"
9620 DATA "PATATAS FRITAS","CHIP
S","FILETE","STEAK","SOPA","SOUP
","HAMBURGUESA","HAMBURGER"
9630 DATA "CALIENTE","HOT","FRIO
","COLD","TODO EL MUNDO","EVERYB
ODY"
9640 DATA "MUSICA","MUSIC","CHIC
O","BOY","CHICA","GIRL","HOMBRE"
,"MAN","MUJER","WOMAN","POR FAVO
R","PLEASE"
9650 DATA "PERRO","DOG","GATO","
CAT","PAJARO","BIRD","CERDO","PI
G","OVEJA","SHEEP"
9660 DATA "HOLA","HELLO","BUENOS
```

```
","GOOD","DIAS","MORNING","NOCHE
S","NIGHT","ADIOS","GOODBYE"
9670 DATA "PUERTA","DOOR","CASA"
,"HOUSE","AMOR","LOVE","SOL","SU
N","LUNA","MOON","POLICIA","POLI
CE"
9680 DATA "CANCION","SONG","AMIG
OS","FRIENDS","MAR"."SEA","CIGAR
RILLOS","CIGARETTES"
9690 DATA "PADRE","FATHER","MADR
E","MOTHER","HIJO","SON","HIJA",
"DAUGHTER"
9999 DATA "*","*"
```

HANSHAN

```
  10 REM ** HANSHAN **
  15 CLS : CLEAR : BRIGHT 1: FLA
SH 0: INK 7: PAPER 1: BRIGHT 0:
BORDER 1
  20 GO SUB 250: REM INICIALIZAC
ION
  30 REM ELECCION DEL PATRON
  40 LET R=INT (RND*3)+1
  50 GO SUB R*50+40
  60 LET KK=RND: IF KK>.65 THEN
 LET DD=RND*20-5: FOR G=DD TO DD
+5: BEEP .1,3: BEEP .1,7: BEEP .
1,-4: BEEP .1,G/2: NEXT G
  65 IF KK<.4 THEN  FOR G=1 TO 5
: BEEP .1,-7: BEEP .1,G*3: BEEP
.2,-2: BEEP .5,3: NEXT G
  68 IF RND<.7 THEN  FOR G=-5 TO
 5: BEEP .1,G: NEXT G
  70 PRINT ''
  80 GO TO 40
  90 REM ** PATRON UNO **
 100 PRINT W$(INT (RND*20)+1);".
..";W$(INT (RND*20)+1)
 110 PRINT TAB 5;"....";W$(INT (
RND*20)+1)
 120 PRINT TAB 8;S$(INT (RND*20)
+1)
 130 RETURN
 140 REM ** PATRON DOS **
 150 PRINT S$(INT (RND*20)+1)
 160 PRINT TAB 3;S$(INT (RND*20)
+1);"..."
 170 PRINT TAB 6;S$(INT (RND*20)
+1)
 180 RETURN
 190 REM ** PATRON TRES **
 200 PRINT TAB 3;W$(INT (RND*20)
+1)
 210 PRINT S$(INT (RND*20)+1)
 220 PRINT TAB 3;W$(INT (RND*20)
+1);", ";S$(INT (RND*20)+1)
 230 RETURN
 240 REM *****************
 250 REM * INICIALIZACION *
 260 CLS
 270 RANDOMIZE
 280 DIM W$(20,12): DIM S$(20,26
)
 290 FOR J=1 TO 20
 300 READ W$(J)
 310 NEXT J
 320 FOR J=1 TO 20
 330 READ S$(J)
 340 NEXT J
 350 RETURN
 360 REM ****** DATOS *******
 370 REM * PALABRAS SUELTAS *
 380 DATA "ESCABULLENDOSE","PISA
NDO","CONTEMPLANDO","MARCHITO","
ESCULPIDO"
 390 DATA "ESCONDIDO","CONFINADO
","ATORMENTADO","DOBLANDO","RETO
RCIENDO"
 400 DATA "GOLPEADO","COLGANDO",
"ENVOLVIENDO","TRANSPARENTE","CA
NSADO"
 410 DATA "HACIA MI","CASCADA","
SACRIFICADOR","RESBALADIZO","EN
PEDAZOS"
 420 REM * FRASES CORTAS *
 430 DATA "EN LA FRIA CORRIENTE"
 440 DATA "AJENO A LA BELLEZA"
 450 DATA "OLAS TRANQUILAS"
 460 DATA "FUERA, DESDE EL ABISM
O"
 470 DATA "SOMBRIO, SOMBRIO"
 480 DATA "INMERSO EN LA OSCURID
AD"
 490 DATA "COGI TUS POEMAS"
 500 DATA "APAGUE LA LAMPARA"
 510 DATA "SE ME CIERRAN LOS OJO
S"
 520 DATA "AQUELLOS LEJANOS DIAS
"
 530 DATA "LOS HOMBRES DE CIENCI
A"
 540 DATA "LOS HOMBRES DE ACCION
"
 550 DATA "ME APRESURO"
 560 DATA "POR QUE DESPERDICIARI
AS"
 570 DATA "CUANDO NOS ENCONTRARE
MOS?"
 580 DATA "DURMIENDO UN POCO"
 590 DATA "Y CON MUCHA PENA"
 600 DATA "POR ESTOS POCOS PASOS
"
 610 DATA "AHORA, AL ANOCHECER"
 620  DATA "HICE PROVECHOSAMENTE
"
```

GRAN ESFUERZO

```
  10 REM ** GRAN ESFUERZO **
  15 PAPER 1: INK 7: BRIGHT 0: F
LASH 0: BORDER 1
  20 POKE 2358,200: CLS
  30 PRINT "QUIERO QUE PIENSES E
N UN HOMBRE,UN CABALLO O UN GORR
ION"
  50 PAUSE 400
  60 PRINT ''
  70 GO SUB 170: REM HACE LAS PR
EGUNTAS
  80 PRINT '; INK 2;"-----------
---------------------"
 100 PRINT '"PULSE ""ENTER"" PAR
A REPETIR       O CUALQUIER TECLA
 Y DESPUES      ""ENTER"" PARA FI
NALIZAR"
 110 INPUT Q$
 120 IF Q$<>"" THEN  STOP
 140 CLS
 150 GO TO 30
 160 REM **********************
 170 REM * HACE LAS PREGUNTAS *
 180 LET CONTADOR=0
 190 PRINT  INK 5;"TIENE DOS PAT
AS O DOS PIERNAS"
 200 GO SUB 310
 210 PRINT  INK 5;"PUEDE CAMINAR
"
 220 GO SUB 310
 230 PRINT  INK 5;"PUEDE VOLAR"
 240 GO SUB 310
 250 PRINT "ESTABAS PENSANDO EN
UN ";
 260 IF CONTADOR=1 THEN  PRINT
FLASH 1; INK 4;"CABALLO"
 270 IF CONTADOR=2 THEN  PRINT
FLASH 1; INK 4;"HOMBRE"
 280 IF CONTADOR=3 THEN  PRINT
FLASH 1; INK 4;"GORRION"
 290 RETURN
 300 REM ***********************
 310 REM SE PROCESA LA RESPUESTA
 320 INPUT "    'S' O 'N'?   ";Z
$
 330 IF Z$<>"N" AND Z$<>"S" THEN
  GO TO 320
 340 IF Z$="S" THEN  LET CONTADO
R=CONTADOR+1
 350 PRINT
 360 RETURN
```

GRAN ESFUERZO-X

```
  10 REM ** GRAN ESFUERZO-X **
  15 POKE 23658,200: INVERSE 0:
FLASH 0: BRIGHT 1: PAPER 1: CLS
  20 GO SUB 940: REM INICIALIZAC
ION
  30 GO SUB 450: REM GANAR EXPER
IENCIA
  40 GO SUB 120: REM DEMOSTRAR E
XPERIENCIA
  50  GO SUB 1060
  60 PRINT "PULSE ""ENTER"" PARA
 OTRA VEZ     O CUALQUIER OTRA TE
CLA Y DESPUES ""ENTER"" PARA ABA
NDONAR."
  80 INPUT Q$
  90 IF Q$="" THEN  GO TO 40
 100 STOP
 110 REM ***********************
 120 REM *DEMOSTRAR EXPERIENCIA*
 130 CLS
 140 GO SUB 1060
 150 PRINT  INK 6; INVERSE 1;"PI
ENSE EN UNO DE LOS SIGUIENTES EL
EMENTOS- "'
 160 FOR J=1 TO OUTCOMES
 170 PRINT TAB J+2;
 180 IF J=OUTCOMES THEN  PRINT "
O ";
 190 PRINT A$(J)
 200 NEXT J
 210 GO SUB 1060
 220 LET RESULT=0
 230 LET X=.5
 240 PRINT "ENTRE ""SI"" O ""NO"
"..."
 250 FOR J=1 TO FACTORS
 260 LET X=X+X
 270 GO SUB 1060
 280 PRINT B$(J)
 290 INPUT E$
 300 IF E$="NO" THEN  LET RESULT
=RESULT+X
 310 NEXT J
 320 PRINT TAB 3; INK 3;">MI RES
ULTADO ERA ";RESULT
 330 GO SUB 1060
 340 LET M=0
 350 LET M=M+1
 360 IF D(M)=RESULT THEN  GO TO
400
 370 IF M<OUTCOMES THEN  GO TO 3
50
 380 PRINT TAB 2; FLASH 1; INK 5
;">>NO PUEDO IDENTIFICARLO<<"; F
LASH 0: FOR R=5 TO 10: BEEP .2,-
4: BEEP .1,10-R: BEEP .1,-5: BEE
P .1,-2: NEXT R
 390 RETURN
 400 PRINT TAB 5;"> ESTABAS PENS
ANDO"
 405 FOR R=5 TO 10: BEEP .1,3: B
EEP .3,-3: BEEP .1,R+5: NEXT R:
BEEP .3,20
 410 PRINT TAB 6;" EN UN ";A$(M)
 420 GO TO 390
 430 RETURN
 440 REM *********************
 450 REM ** LLENAR MATRICES **
 460 PRINT TAB (20-LEN N$/2);N$
 470 GO SUB 1060
 480 REM NOMBRE DE LOS ELEMENTOS
 490 FOR J=1 TO OUTCOMES
 500 GO SUB 1060
 510 PRINT "CUAL ES EL ELEMENTO
";J;" ";
 520 INPUT A$(J): PRINT  INK 4;A
$(J)
 530 NEXT J
 540 CLS
 550 REM COGER CUESTIONES PARA P
REGUNTARLAS
 560 FOR J=1 TO FACTORS
 570 GO SUB 1060
 580 PRINT "ENTRE LA PREGUNTA ";
J
 590 INPUT B$(J)
 600 NEXT J
 610 CLS
 620 REM ** ADQUIRIR EXPERIENCIA
 **
 630 FOR J=1 TO OUTCOMES
 640 CLS
 650 GO SUB 1060
 660 PRINT "RESPONDA A LA SIGUIE
NTE PREGUNTA"'"PARA EL ELEMENTO
>";A$(J);"<"
 680 GO SUB 1060
 690 PRINT "ENTRE ""SI"" O ""NO"
""
 700 LET X=.5
 720 FOR K=1 TO FACTORS
 730 LET X=X+X
 740 GO SUB 1060
 750 PRINT TAB 4;"> ";B$(K);
 760 LET MULTI=0
 770 INPUT Y$
 780 IF Y$="NO" THEN  LET MULTI=
1
```

```
 790 LET D(J)=D(J)+X*MULTI
 800 NEXT K
 810 NEXT J
 820 CLS
 830 PRINT "ESTA ES MI EXPERTA B
ASE:"
 840 FOR J=1 TO OUTCOMES
 850 GO SUB 1060
 860 PRINT A$(J);" ---";D(J)
 870 NEXT J
 880 GO SUB 1060
 890 PRINT TAB 8;"PULSE ""ENTER"
""
 900 INPUT Q$
 910 CLS
 920 RETURN
 930 REM ********************
 940 REM ** INICIALIZACION **
 950 CLS
 960 INPUT "NOMBRE DEL SISTEMA?
";N$
 970 GO SUB 1060
 980 INPUT "NUMERO DE ELEMENTOS?
 ";OUTCOMES
 990 GO SUB 1060
1000 INPUT "NUMERO DE FACTORES A
 CONSIDERAR? ";FACTORS
1010 DIM A$(OUTCOMES,15): DIM B$
(FACTORS,30)
1020 DIM D(OUTCOMES)
1030 CLS
1040 RETURN
1050 REM ********************
1060 PRINT : PRINT
1070 RETURN
```

ELECCION DE UN MICROPROCESADOR

```
  10 REM **   ELECCION DE UN   **
         **   MICROPROCESADOR **
  15 POKE 23658,200: FLASH 0: BR
IGHT 1: INK 7: PAPER 1: BRIGHT 0
: BORDER 1: CLS
  20 GO SUB 580: REM INICIALIZAC
ION
  30 GO SUB 450: REM MUESTRA EL
CONTENIDO DE LA BASE DE DATOS
  40 GO SUB 120: REM IDENTIFICA
EL MICROPROCESADOR
  50 GO SUB 720
  60 PRINT "PULSE ""ENTER"" PARA
 OTRO MICROPROCESADOR, O"
  70 PRINT "CUALQUIER OTRA TECLA
 Y LUEGO ""ENTER"" PARA FINALIZA
R"
  80 INPUT Q$
  90 IF Q$="" THEN  GO TO 40
 100 STOP
 110 REM ***********************
 120 REM IDENTIFICACION DEL MICR
OPROCESADOR
 130 CLS
 140 GO SUB 720
 150 PRINT "PUEDO IDENTIFICAR LO
S SIGUIENTES MICROPROCESADORES"
 160 FOR J=1 TO ELEMENTOS
 170 PRINT TAB J+2
 180 IF J=ELEMENTOS THEN  PRINT
"O "
 190 PRINT  INK 4;A$(J)
 200 NEXT J
 210 GO SUB 720
 220 LET RESULT=0
 230 LET X=.5
 240 PRINT "POR FAVOR INTRODUZCA
 ""S"" O ""N"""
 250 FOR J=1 TO FACTORS
 260 LET X=X+X
 280 PRINT 'B$(J)
 290 INPUT E$
 295 IF E$<>"S" AND E$<>"N" THEN
  GO TO 290
 300 IF E$<>"N" THEN  LET RESULT
=RESULT+X
 310 NEXT J
 320 PRINT TAB 3; INK 3;"> MI RE
SULTADO FUE ";RESULT
 330 GO SUB 720
 340 LET M=0
 350 LET M=M+1
 360 IF D(M)=RESULT THEN  GO TO
400
 370 IF M<ELEMENTOS THEN  GO TO
350
 380 FOR R=1 TO 5: BEEP .1,R: BE
EP .2,R+10: NEXT R: PRINT TAB 3;
 INK 5; FLASH 1;">NO PUEDO IDENT
IFICARLO"; FLASH 0
 390 RETURN
 400 BEEP 1,-10: FOR R=1 TO 3: B
EEP .2,6: BEEP .1,3: BEEP .1,2:
NEXT R: BEEP .5,7: PRINT TAB 5;"
EL QUE TU TIENES"'TAB 8;"ES ";A$
(M)
 420 GO TO 390
 430 RETURN
 440 REM ***********************
 450 REM ** MUESTRA EL CONTENIDO
 DE LA BASE DE DATOS **
 460 CLS
 470 PRINT "ESTA ES MI BASE DE D
ATOS:"
 480 FOR J=1 TO ELEMENTOS
 490 GO SUB 720
 500 PRINT A$(J);" ---";D(J)
 510 NEXT J
 520 GO SUB 720
 530 PRINT TAB 8;"PULSA LA TECLA
 ""ENTER"""
 540 INPUT Q$
 550 CLS
 560 RETURN
 570 REM *******************
 580 REM ** INICIALIZACION **
 590 CLS
 600 RESTORE
 610 LET ELEMENTOS=6
 620 LET FACTORS=6
 630 DIM A$(ELEMENTOS,17): DIM B
$(FACTORS,60): DIM D(ELEMENTOS)
 640 FOR J=1 TO ELEMENTOS
 650 READ A$(J),D(J)
 660 NEXT J
 670 FOR J=1 TO FACTORS
 680 READ B$(J)
 690 NEXT J
 700 RETURN
 710 REM ***********
 720 PRINT : PRINT
 730 RETURN
```

```
 740 REM ***********************
 750 REM NOMBRE DEL MICROPROCESA
DOR
 760 DATA "TMS 9940 (NMOS)",44,"
68000 (NMOS)",12
 770 DATA "9940 (I3L)",56,"MN161
0 (NMOS)",46
 780 DATA "8086",60,"Z8001",28
 790 REM ** PREGUNTAS **
 800 DATA "ES LA LONGITUD DE PAL
ABRA DE 32 BITS"
 810 DATA "TIENE 64K DE MEMORIA"
 820 DATA "ES LA FRECUENCIA DEL
OSCILADOR DE 5 MHz O MENOR"
 830 DATA "EL TIEMPO DE EJECUCIO
N MINIMO ES DE 3 MICROSEGUNDOS O
 MENOS"
 840 DATA "TIENE EL JUEGO DE INS
TRUCCIONES MAS DE 71"
 850 DATA "TIENE EL INTEGRADO 40
 PATILLAS"
```

APRENDE-SOLO

```
  10 REM ** APRENDE-SOLO **
  15 BRIGHT 1: FLASH 0: INK 7: B
ORDER 1: BRIGHT 0: PAPER 1: CLS
  20 GO SUB 400: REM INICIALIZAC
ION
  30 REM *BUCLE MAYOR DE APRENDI
ZAJE*
  40 CLS
  50 FOR J=1 TO FACT
  60 LET C(J)=0
  70 NEXT J
  80 PRINT
  90 GO SUB 130
 100 GO TO 40
 110 REM *******************
 120 REM *** DEMOSTRACION ***
 130 PRINT  INK 5;"Ahora demostr
are mi experiencia..."
 140 PRINT '"Piense en uno de lo
s elementos"
 145 PRINT
 150 FOR J=1 TO FACT
 170 PRINT  INK 5;"Es ";B$(J); I
NK 7; BRIGHT 1;" Verdadero?( ""S
I"" o ""NO"" )"
 180 INPUT Z$
 190 IF Z$<>"SI" AND Z$<>"NO" TH
EN  GO TO 180
 200 IF Z$="SI" THEN  LET C(J)=1
 210 PRINT TAB 29; INK 4;C(J)
 220 NEXT J
 230 LET BRAYN=0
 240 FOR J=1 TO FACT
 250 LET BRAYN=BRAYN+C(J)*D(J)
 260 NEXT J
 270 PRINT TAB 20;"> CEREBRO=";B
RAYN
 280 IF BRAYN>=0 THEN  PRINT "El
 elemento es ";A$(1): LET EX=-1
 290 IF BRAYN<0 THEN  PRINT "El
elemento es ";A$(2): LET EX=1
 300 PRINT "Es esto correcto? ("
"SI"" o ""NO"")"
 310 INPUT Z$
 320 IF Z$<>"SI" AND Z$<>"NO" TH
EN  GO TO 310
 330 PRINT
 340 IF Z$="SI" THEN  FOR R=1 TO
 5: BEEP .1,R: NEXT R: GO TO 380
 345 FOR R=-1 TO -5 STEP -1: BEE
P .1,R: NEXT R
 350 FOR J=1 TO FACT
 360 LET D(J)=D(J)+EX*C(J)
 370 NEXT J
 380 RETURN
 390 REM *******************
 400 REM ** INICIALIZACION **
 410 CLS
 420 LET OTCO=2: REM NUMERO DE E
LEMENTOS
 430 PRINT : PRINT
 440 INPUT "Cuantos factores ";F
ACT
 450 DIM A$(OTCO,30): REM NUMERO
 DE ELEMENTOS
 460 DIM B$(FACT,30): REM NOMBRE
 DE LOS FACTORES
 470 DIM C(FACT): DIM D(FACT)
 480 CLS
 490 FOR J=1 TO FACT
 510 PRINT '"Entra el factor ";J
;"";
 520 INPUT B$(J): PRINT  INK 4;B
$(J)
 530 NEXT J
 550 CLS
 560 FOR J=1 TO OTCO
 580 PRINT '"Entra el elemento "
;J;"";
 590 INPUT A$(J): PRINT  INK 5;A
$(J)
 600 NEXT J
 610 RETURN
```

MULTI-APRENDE-SOLO

```
  10 REM * MULTI-APRENDE-SOLO *
  15 INVERSE 0: POKE 23658,200:
BRIGHT 1: FLASH 0: PAPER 1: INK
7: BRIGHT 0: BORDER 1: CLS
  20 GO SUB 550: REM INICIALIZAC
ION
  30 CLS
  40 PRINT "ESTOS SON LOS POSIBL
ES ELEMENTOS:"
  50 PRINT
  60 FOR J=1 TO OCTO
  70 PRINT TAB J;A$(J)
  80 NEXT J
  90 PRINT
 100 PRINT "POR FAVOR PIENSE EN
UNO DE ELLOS"
 110 PRINT
 120 PRINT "PULSE"; FLASH 1;" EN
TER"; FLASH 0;" CUANDO ESTE PREP
ARADO"
 130 INPUT Z$
 140 CLS
 150 PRINT "POR FAVOR RESPONDA '
S' O 'N' A CADA UNA"
 160 PRINT "DE LAS SIGUIENTES PR
EGUNTAS,RELACIONADAS"
 170 PRINT "CON EL ELEMENTO EN Q
UE PIENSO."
 180 PRINT
 190 LET CONTADOR=0
 200 LET X=.5
 210 FOR J=1 TO FACT
 220 LET X=X+X
 230 PRINT F$(J)
 240 INPUT Z$: PRINT "? "; FLASH
 1;Z$''
 250 IF Z$<>"S" AND Z$<>"N" THEN
 GO TO 240
 260 IF Z$="S" THEN  LET CONTADO
R=CONTADOR+X
 270 NEXT J
 280 LET X=0
 290 FOR J=1 TO OCTO
 300 IF CONTADOR=B(J) THEN  LET
X=J
 310 NEXT J
 320 IF X<>0 THEN  GO TO 410
 330 LET X=INT (RND*OCTO)+1
 340 REM ** RECHAZA TODAS LAS RE
SPUESTAS INCORRECTAS **
 350 LET FLAG=0
 360 FOR J=1 TO OCTO
 370 IF B(J)=0 THEN  GO TO 390
 380 IF X=J AND CONTADOR<>B(J) T
HEN  LET FLAG=1
 390 NEXT J
 400 IF FLAG=1 THEN  GO TO 330
 410 PRINT "ESTABAS PENSANDO EN
";A$(X)
 420 PRINT
 430 PRINT "INTRODUZCA 'S' SI TE
NGO RAZON,Y 'N' EN CASO CONTRARI
O"
 440 INPUT Z$
 450 IF Z$<>"S" AND Z$<>"N" THEN
 GO TO 440
 460 IF Z$="S" THEN  LET B(X)=CO
NTADOR: FOR R=1 TO 5: BEEP .1,2:
 BEEP .1,5: BEEP .1,R: GO TO 30
 470 FOR R=1 TO 4: BEEP .1,7: BE
EP .2,8: BEEP .1,R*2: NEXT R: PR
INT "CUAL DEBERIA HABER SIDO LA
RESPUESTA?"
 480 INPUT Z$
 490 FOR J=1 TO OCTO
 500 IF A$(J, TO LEN Z$)=Z$ THEN
 LET B(J)=CONTADOR
 510 NEXT J
 520 GO TO 30
 530 STOP
 540 REM **************
 550 REM INICIALIZACION
 560 CLS
 570 PRINT "CUANTOS ELEMENTOS? "
;: INPUT OCTO
 580 PRINT  INK 5;OCTO''
 590 PRINT "CUANTOS FACTORES? ";
: INPUT FACT
 600 PRINT  INK 5;FACT''
 610 LET X=OCTO+FACT
 620 DIM A$(OCTO,30)
 630 DIM F$(FACT,30): DIM B(X)
 640 FOR J=1 TO OCTO
 650 PRINT "POR FAVOR INTRODUZCA
 EL NOMBRE DEL ELEMENTO";J
 660 INPUT A$(J): PRINT  INK 4;"
?";A$(J)
 670 NEXT J
 680 CLS
 690 FOR J=1 TO FACT
 700 PRINT "POR FAVOR INTRODUZCA
 EL NOMBRE DEL FACTOR ";J
 710 INPUT F$(J): PRINT  INK 4;"
?";F$(J)
 720 NEXT J
 730 RETURN
```

MULTI-APRENDE-SOLO VERSION DOS

```
 10 REM * MULTI-APRENDE-SOLO *
       **   VERSION  DOS    **
 15 BRIGHT 1: FLASH 0: PAPER 1:
INK 7: BRIGHT 0: BORDER 1: CLS
 20 GO SUB 550: REM INICIALIZAC
ION
 30 CLS
 40 PRINT  INK 3;'"ESTOS SON LO
S POSIBLES ELEMENTOS:"
 50 PRINT
 60 FOR J=1 TO OCTO
 70 PRINT TAB J; INK 4;A$(J)
 80 NEXT J
 90 PRINT
 100 PRINT "POR FAVOR PIENSE EN
UNO DE ELLOS"
 110 PRINT
 120  PRINT "PULSA "; FLASH 1; I
NK 3;"ENTER"; INK 7; FLASH 0;" C
UANDO ESTES PREPARADO"
 130 INPUT Z$
 140 CLS
 150 PRINT "POR FAVOR RESPONDA '
S' O 'N' A CADA UNA"
 160 PRINT "DE LAS SIGUIENTES PR
EGUNTAS,RELACIONADAS"
 170 PRINT "CON EL ELEMENTO EN Q
UE PIENSO"
 180 PRINT
 190 LET CONTADOR=0
 200 LET X=.5
 210 FOR J=1 TO FACT
 220 LET X=X+X
 230 PRINT  BRIGHT 1;F$(J)
 240 INPUT Z$: PRINT "? "; FLASH
 1;Z$''
 250 IF Z$<>"S" AND Z$<>"N" THEN
  GO TO 240
 260 IF Z$="S" THEN  LET CONTADO
R=CONTADOR+X
 270 NEXT J
 280 LET X=0
 290 FOR J=1 TO OCTO
 300 IF CONTADOR=B(J) THEN  LET
X=J
 310 NEXT J
 320 IF X<>0 THEN  GO TO 410
 330 LET X=INT (RND*OCTO)+1
 340 REM **RECHAZA TODAS LAS RES
PUESTAS INCORRECTAS **
 350 LET FLAG=0
 360 FOR J=1 TO OCTO
 370 IF B(J)=0 THEN  GO TO 390
 380 IF X=J AND CONTADOR<>B(J) T
HEN  LET FLAG=1
 385 IF C(X)=CONTADOR OR D(X)=CO
NTADOR THEN  LET FLAG=1
 390 NEXT J
 400 IF FLAG=1 THEN  GO TO 330
 410 PRINT "ESTABAS PENSANDO EN
";A$(X)
 420 PRINT
 430 PRINT "INTRODUZCA 'S' SI TE
NGO RAZON, Y 'N' EN CASO CONTRAR
IO"
 440 INPUT Z$
 450 IF Z$<>"S" AND Z$<>"N" THEN
  GO TO 440
 460 IF Z$="S" THEN  LET B(X)=CO
NTADOR: FOR R=1 TO 5: BEEP .1,2:
 BEEP .1,5: BEEP .1,R: GO TO 30
 470 IF C(X)=0 THEN  LET C(X)=CO
NTADOR: GO TO 30 .
 520 LET D(X)=CONTADOR: GO TO 30
 530 STOP
 540 REM *************
 550 REM INICIALIZACION
 560 CLS
 570 PRINT "CUANTOS ELEMENTOS? "
;: INPUT OCTO
 580 PRINT  INK 5;OCTO''
 590 PRINT "CUANTOS FACTORES? ";
: INPUT FACT
 600 PRINT  INK 5;FACT''
 610 LET X=OCTO+FACT
 620 DIM A$(OCTO,30)
 630 DIM F$(FACT,30): DIM B(X):
DIM C(X): DIM D(X)
 640 FOR J=1 TO OCTO
 650 PRINT "POR FAVOR INTRODUZCA
 EL NOMBRE DEL ELEMENTO ";J
 660 INPUT A$(J): PRINT  INK 4;"
?";A$(J)
 670 NEXT J
 680 CLS
 690 FOR J=1 TO FACT
 700 PRINT "POR FAVOR INTRODUZCA
 EL NOMBRE DEL FACTOR ";J
 710 INPUT F$(J): PRINT  INK 4;"
?";F$(J)
 720 NEXT J
 730 RETURN
```

ANAYA MULTIMEDIA

Colección «MICROINFORMATICA»

PROGRAMACION AVANZADA EN BASIC.—Bishop, P.
PASCAL A PARTIR DEL BASIC.—Brown, P.
EL ORDENADOR PERSONAL: COMO ELEGIRLO Y UTILIZARLO.—Cavalcoli, A.
PROGRAMACION EN BASIC: UN METODO PRACTICO.—Dachslager, H.; Hayashi, M.; Zucker, R.
TU PRIMER LIBRO DEL ZX SPECTRUM.—Dewhirst, J.; Tennison, R.
ASTRONOMIA: EL UNIVERSO EN TU ORDENADOR (ZX Spectrum).—Gavin, M.
EL ORDENADOR Y TUS HIJOS.—Hammond, R.
EL LIBRO GIGANTE DE LOS JUEGOS PARA ORDENADOR.—Hartnell, T.
BITS Y BYTES: INICIACION A LA INFORMATICA.—Heller, R. S.; Martin, C. D.
MICROINFORMATICA. Conceptos básicos.—Hollerbach, L.
DESCUBRE LAS MATEMATICAS CON TU MICRO.—Johnson, D.
EL ORDENADOR EN EL AULA.—Pentiraro, E.
EL LIBRO DEL BASIC.—Zaks, R.

DISEÑO DE GRAFICOS Y VIDEOJUEGOS (ZX Spectrum).—Angell, I. O.; Jones, B. Y.
MATEMATICAS DIVERTIDAS EN BASIC.—Kosniowski, C.
INTELIGENCIA ARTIFICIAL. CONCEPTOS Y PROGRAMAS.—Hartnell, T.
¿QUE ES LA TELEMATICA? Nuevas tecnologías en la sociedad de la información.—Servello, F.
COMO SE PROGRAMAN LOS ORDENADORES. Programación estructurada básica.—De Rosso, V.
«SPRITES» Y GRAFICOS EN LENGUAJE MAQUINA (ZX Spectrum).—Durst, J.
EL LIBRO GIGANTE DE LOS JUEGOS PARA ZX SPECTRUM.—Hartnell, T.
LENGUAJE MAQUINA AVANZADO (ZX Spectrum).—Webb, D.
LOS ORDENADORES NO MUERDEN.—Coccione, L.; Winter, G.

NOTAS

NOTAS

La Inteligencia Artificial es el estudio de la inteligencia humana usando los métodos de la Informática. Su objetivo es lograr que los ordenadores se comporten como si pensasen.

INTELIGENCIA ARTIFICIAL. CONCEPTOS Y PROGRAMAS, investiga el mundo apasionante de la inteligencia artificial con algunos de los ejemplos más famosos:

SILOGISMOS:	Un programa que razona.
TICTAC:	Un juego de TRES EN RAYA que aprende a medida que juega.
BLOQUELANDIA:	Un programa que se comunica con nosotros en castellano.
DOCTOR:	La versión BASIC más sofisticada que se ha publicado nunca del famosísimo ELIZA.
TRADUCTOR:	Un programa que traduce del español al «spanglish».
HANSHAW:	Un poeta automático.
APRENDE-SOLO:	Un sistema experto que aprende.

INTELIGENCIA ARTIFICIAL. CONCEPTOS Y PROGRAMAS, te lanza a una aventura fascinante en un reino donde se difuminan las barreras entre el hecho científico y la ciencia ficción.

¿Puede una máquina pensar realmente?... Acompaña a Tim Hartnell en este apasionante libro hasta encontrar la respuesta.



Inteligencia Artificial

Tim Hartnell


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
https://acpc.me/